A TABU SEARCH METAHEURISTIC FOR THE AIR REFUELING TANKER ASSIGNMENT PROBLEM

THESIS

Shay R. Capehart, First Lieutenant, USAF

AFIT/GOR/ENS/00M-07

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 2000	Master's Thesis

1. **TITLE AND SUBTITLE**

A TABU SEARCH METAHEUSTIC FOR THE AIR REFUELING TANKER ASSIGNMENT PROBLEM

* **FUNDING NUMBERS**

***AUTHOR(S)**

Shay R. Capehart, First Lieutenant, USAF

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 P Street, Building 640
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GOR/ENS/00M-07

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

HQ AMC/XPY
402 Scott Drive, Unit 3L3
Scott AFB, IL 62225-5307
DSN: 576-5954

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. **SUPPLEMENTARY NOTES**

Dr. James T. Moore, AFIT/EN James.Moore@afit.af.mil (937) 255-6565

12a. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

ABSTRACT *(Maximum 200 Words)*

a joint effort between Air Mobility Command (AMC) and the Air Force Institute of Technology, we present a Tanker Assignment Problem (TAP) Tool capable of providing tanker mission plans for deployment scenarios. Due to the complex nature of extracting a mission plan from the Combined Mating and Ranging Planning System (CMARPS), AMC requires a tool to provide similar results in a simpler and less time consuming manner. The tool developed allows AMC to input several receiver groups consisting of various aircraft types and numbers. Each receiver group contains a point of origin and destination, with the option of providing one waypoint along the path. In addition, each group has a ready to load date (RLD) and required delivery date (RDD). The user may also specify the locations of military tanker aircraft. The main goal of this tool is to assign the tankers to the different refueling points of the receiver groups so that all receiver groups arrive before their RDD. Secondary goals include the reuse of tankers and limiting the total flight distance for all tanker aircraft. The TAP Tool uses the heuristic technique tabu search to determine an assignment of tankers to receiver groups during a deployment.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Tabu Search, Heuristics, Metaheuritics, Tanker Scheduling, Assignment Problem	97
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNC

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GOR/ENS/00M-07

A TABU SEARCH METAHEURISTIC FOR THE AIR REFUELING TANKER ASSIGNMENT PROBLEM

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the Degree of

Master of Science in Operations Research

Shay R. Capehart, B.S.

First Lieutenant, USAF

March 2000

AFIT/GOR/ENS/00M-07

A TABU SEARCH METAHEURISTIC FOR THE AIR REFUELING TANKER ASSIGNMENT PROBLEM

Shay R. Capehart, B.S.
First Lieutenant, USAF

Approved:

James T. Moore — 6 Mar 00

Co-Advisor James T. Moore, Ph.D. — date
Associate Professor of Operations Research
Department of Operational Sciences
Air Force Institute of Technology

Raymond Hill — 6 Mar 00

Co-Advisor Raymond Hill, Major, USAF — date
Assistant Professor of Operations Research
Department of Operational Sciences
Air Force Institute of Technology

Acknowledgements

I would like to thank my advisors Dr. Moore and Maj. Hill for patiently reading the many thesis drafts I went through to reach this goal. Further thanks go to the rest of the ENS faculty for providing me with the knowledge to complete the tasks needed for this thesis. The quality of instruction I received from my AFIT classes was far superior to any of the classes I have taken from civilian schools. Thanks go to Dr. Barnes at the University of Texas in Austin for introducing me to the joys of Tabu Search. Without TS, I would still be searching for that optimum answer. Thank you Maj. Ryer and AMC for providing me with the information needed to make my code run accurately.

A special thanks goes to my friends and familiy. To Chris, Paige, Rob, Gaybe, and Shane, you gave me a social life I thought I would miss during graduate school. To Nicholas, you were always there when I needed you. Let's continue to support each other in New Mexico. Finally, to my family. I could not imagine these last 18 months without the support of my parents and Karl. I love you all.

Shay R. Capehart

Table of Contents

List of Figures

List of Tables

Abstract

In a joint effort between Air Mobility Command (AMC) and the Air Force Institute of Technology, we present a Tanker Assignment Problem (TAP) Tool capable of providing tanker mission plans for deployment scenarios. Due to the complex nature of extracting a mission plan from the Combined Mating and Ranging Planning System (CMARPS), AMC requires a tool to provide similar results in a simpler and less time consuming manner. The tool developed allows AMC to input several receiver groups consisting of various aircraft types and numbers. Each receiver group contains a point of origin and destination, with the option of providing one waypoint along the path. In addition, each group has a ready to load date (RLD) and required delivery date (RDD). The user is also able to specify the locations of military tanker aircraft. The main goal of this tool is to assign the tankers to the different refueling points of the receiver groups so that all receiver groups arrive before their RDD. Secondary goals include the reuse of tankers and limiting the total flight distance for all tanker aircraft. The TAP Tool uses the heuristic technique tabu search to determine an assignment of tankers to receiver groups during a deployment.

Chapter 1. Background and Statement of the Problem

1.1 Background

The KC-135 Stratotanker and KC-10 Extender aircraft provide in-flight refueling for numerous aircraft in the Air Force and Navy. These assets provide force enhancement capabilities, most notably extended range and early arrival. For example, with in-flight refueling, fighter aircraft can fly non-stop from the east coast of the United States to Saudi Arabia in 15 hours rather than the 47 hours required by landing enroute to refuel (Hostler 1987). By providing this extended range, the aircraft avoid the enroute landings, thus adding security and safety. A single Department of Defense (DoD) agency, Air Mobility Command (AMC), manages the allocation of these tanker aircraft to refueling requests.

The KC-135 first entered the Air Force inventory in 1956 to extend the range of Strategic Air Command's B-52 strategic bomber fleet (Congressional Budget Office 1985). Today, almost every aircraft in the Air Force and Navy inventory can be refueled while airborne. While the number of aircraft capable of aerial refueling has increased, the number of tanker aircraft has decreased. The current fleet of KC-135 tankers consists of 500 aircraft (300 fewer than 30 years ago), with an additional 50 KC-10 tanker aircraft in the inventory. This fleet provides in-flight refueling to the majority of DoD's military fixed wing aircraft and many military aircraft from other nations. It allows combat aircraft to strike more targets deep in enemy territory, extends the time fighter aircraft can protect friendly forces from attack by enemy aircraft, and supports the extension of the United States' military presence throughout the world.

Due to limited tanker aircraft compared to the number needed during an operational deployment, not all refueling requests can be met simultaneously. This, in turn, causes many receiver groups to arrive at their destinations later than scheduled (receiver groups are formations of similar aircraft; i.e., 4 F-16s or 2 C-130s). Therefore, the number of receiver groups that arrive tardy is largely determined by the allocation of the tanker resources. The allocation of tanker resources an air refueling tanker schedule that assigns tankers to receiver groups.

1.2 Problem Statement

The Air Force's fleet of air refueling tanker aircraft provides essential support for deployment of combat and combat support aircraft in crisis situations. The term "deployment" refers to moving military forces to where the forces are needed so that they are ready to respond to a crisis. Air refueling aircraft provide this support by offloading fuel to receiver aircraft in flight. AMC receives analytical queries from various agencies regarding different planning scenarios. AMC runs computer simulations to answer these questions. Those questions that involve deployment issues and tanker schedules require AMC to make extensive runs with the Combined Mating and Ranging Planning System (CMARPS).

Currently, the CMARPS software tool aids AMC in analyzing, planning, and scheduling deployment of air refueling tankers to support immediate and anticipated military operations. This tool provides actual tanker/receiver aircraft schedules and flight plans that take into account numerous system constraints. However, it can take up to two weeks to run the number of scenarios necessary to allocate the tanker resources for one

deployment. Another drawback of CMARPS is that it is not interoperable with AMC's airlift simulation. Since tanker and airlift missions are interrelated and compete for limited airbase resources, there needs to be some interaction between the two simulation tools. This research focuses on improving AMC's analytical support response.

1.3 Scope

An overall tanker-planning tool may be viewed as a model of tanker deployment and employment operations based on planning factors. Our proposed algorithm addresses the deployment phase of the overall model, while deferring employment planning to future research. Our project focuses on answering the following questions:

1. Given receiver group deployment schedules, receiver aircraft characteristics, and system constraints/capacities, how many tankers will it take to meet receiver deployment air refueling requirements?
2. Given system constraints/capacities and a fixed number of tankers, how quickly will receiver aircraft deploy?

This problem involves non-homogeneous vehicles, located at multiple locations, assigned to receiver groups. In this manner, the problem is an assignment problem. However, time-windows are introduced due to the limited number of tankers and the composition of the receiver groups. The receiver groups consist of fighter and "heavy" aircraft, the former requiring escort across open water by the tanker aircraft, while the latter only need refueling from the tanker aircraft.

Factors that effect this problem include the tanker and receiver aircraft fuel capacities and burn rates, deployment distances, number of aircraft to be supported and

time frames, geographic location of receiver origins, destinations, weather, requirements for tankers to escort receivers, formation size, crew duty limitations, missed refueling, base requirements, and aircraft servicing constraints. The latter three factors are not considered in this research and are left for future research.

1.4 Contribution of Research

This effort provides AMC a quick running tanker-scheduling model. The algorithm proposed in this research can serve as the starting point for a future tanker-scheduling model that is interoperable with an airlift model.

1.5 Report Overview

The remainder of this thesis is organized as following: Chapter 2 reviews the literature pertaining to this research. Chapter 3 presents a proposed methodology for conducting the research. Chapter 4 presents the results of the research. Finally, Chapter 5 discusses issues for further research and improvements that can be incorporated for future research on this topic.

Chapter 2. Literature Review

2.1 Tanker Scheduling Tools

2.1.1 Combined Mating And Ranging Planning System (CMARPS)

CMARPS is the current tool used by AMC to determine when, where, and how much air refueling is required for mission aircraft. Originally introduced in 1982, CMARPS noe functions with several other AMC tools providing optimized assignment of tanker resources to meet the air refueling requirements. CMARPS considers single cells of similar aircraft types. It determines the location and time for each cell's refueling points along the path. An effort is made to minimize the tanker aircraft and sorties flown for each mission. Finally, CMARPS assigns specific tankers to the refueling points.

The mission routing determines the fuel requirements for the receiver group. CMARPS routes the mission considering the following criteria: avoid restricted airspace, minimize threat exposure, deconflict routes in strike zone, and satisfy time over targets. Once the fuel requirements are determined, CMARPS assigns tankers to meet the requirements. The considerations during this phase include: minimize use of tanker resources, minimize tanker fuel consumption, use air refuelable tankers, regenerate tankers for tanker reuse, and meet the abort base requirements. Once the tanker assignments are made, CMARPS simulates the aircraft mission using formulas for winds and fuel consumptions to generate a final mission schedule and flight plan (LOGICON 1996).

2.1.2 Quick Look Tool for Tanker Deployment

Russina, Ruthsatz, and Russ (1999) provide a prototype tool to evaluate tanker allocation for the aircraft deployment mission. Their Quick Look Tool makes basic assumptions regarding aircraft capabilities and interactions between receiver groups and tankers. The Quick Look Tool functions as a relatively simple tool for modeling and predicting air refueling tanker capabilities for supporting deployment of combat and combat support aircraft. The AMC tanker-scheduling problem involves a wide scope of system variables, constraints, and potential analysis areas. The example provided by Russina, Ruthsatz, and Russ presents an in-depth explanation of the deployment issues associated with tanker deployment.

Example Deployment

Figure 1 dipicts five airbases supporting two different deployment operations: (i) OP1 represents the movement of a single or multiple receiver aircraft from initial position *Base 5* to desired final position *Base1*; and (ii) OP2 represents the movement of a single or multiple receiver aircraft from initial position *Base 2* to desired final position *Base4*.

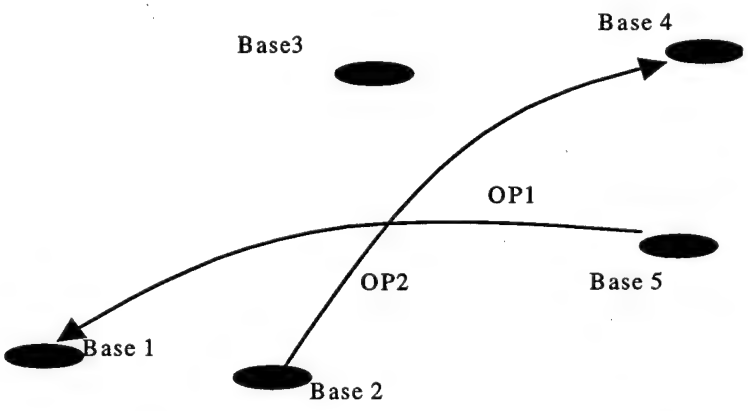

Figure 1: Example deployment flight paths

Both operations are subject to the schedule in Table 1.

Table 1: Example deployment schedule

Deployment Operation	Receiver Origin	Receiver Destination	ALD	RDD
OP1	Base 5	Base 1	1	3
OP2	Base 2	Base 4	2	4

As shown in the columns of the table, the aircraft involved in deployment OP1 have an available to load date (ALD) from *Base 5* on the first day of the designated scheduling period and have a required delivery date (RDD) at *Base 1* on the third day. Additionally, the aircraft involved in deployment OP2 are available to depart on the second day of the scheduling period and need to arrive at their destination by the fourth day. The aircraft involved in these missions may not have enough onboard fuel to complete the desired deployment, so they need support from the refueling tankers located at any one of the five bases in Figure 1.

Several different objectives exist for this problem. The primary objectives include meeting scheduling goals, minimizing overall aircraft fuel consumption, and minimizing the number of tankers used. Each objective is similarily constrained by the interaction of receivers and tankers within the system, and the individual fuel consumption characteristics of each aircraft.

Receiver and Tanker Interaction

Several factors must be explicitly modeled when planning the deployment:

Distance = Rate x Time. Travel distance, time, and air speed rate are clearly key factors in scheduling tankers to support the movement of receiver aircraft. For the

problem at hand, the only known value in this equation is the distance between origin and destination airbases. Increasing the rate at which receivers travel improves solutions to the scheduling problem, but increases fuel consumption rates, affecting fuel consumption goals. Varying receiver travel rates enroute drastically increases the complexity of the scheduling problem, which generally assumes constant travel rates. Thus, a well-defined protocol for deciding travel rates and times must be determined before embarking on analysis of scheduling options.

Geography. Another issue of importance is the geographical positioning of airbases involved, with respect to both the distance between, and the relative positioning, of particular bases. Distance between bases is a critical factor in the scheduling problem. In addition to aircraft airspeed, altitude, gross weight, wind conditions, and refueling speeds, the distance of flight drives a receiver's total fuel consumption, and thus the amount of fuel required to complete the desired mission.

Relative position of airbases is important. The relative positioning of airbases affects the influence of wind on aircraft ground speeds and the distance of available tankers from receiver routes of flight. This can force the alteration of receiver and tanker flight paths due to inaccessible airspace. Thus, developing appropriate assumptions to model the geographical situation is critical to the accuracy of the scheduling tool.

Escort Requirements and Refueling Points. Safety regulations require the Air Force escort fighter aircraft with refueling tankers when traveling over large bodies of water. Heavier aircraft, such as cargo planes and bombers, do not require tanker escort. Consider the example deployment problem and suppose that OP1 involves movement of an F-16 fighter from origin Base 5 to destination Base 1. In addition, assume origin Base

5 is Travis Air Force Base and the destination Base 1 is Kunsan Air Base. If the F-16 fighter must traverse the Pacific Ocean to complete the OP1 operation, at least one available tanker must rendezvous and accompany the fighter by the time it begins to fly over open water (see Figure 2).

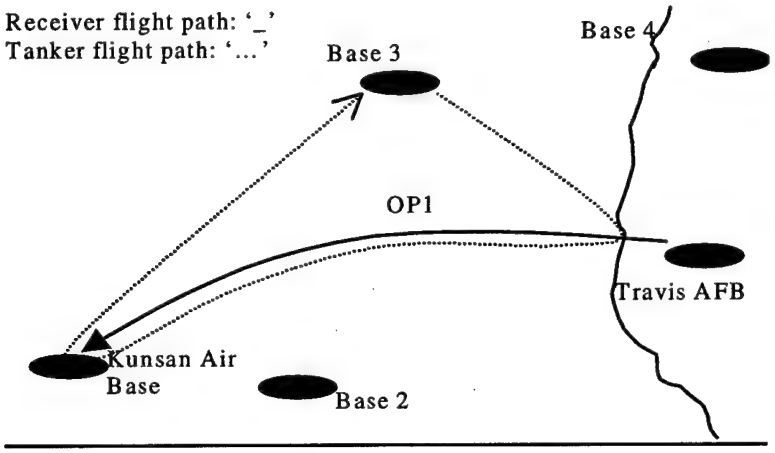

Figure 2: Tanker escort of an F-16 fighter

On the other hand, if the aircraft is a C-17 cargo plane, the tanker only needs to rendezvous with the receiver for as long as it takes to refuel (see Figure 3).

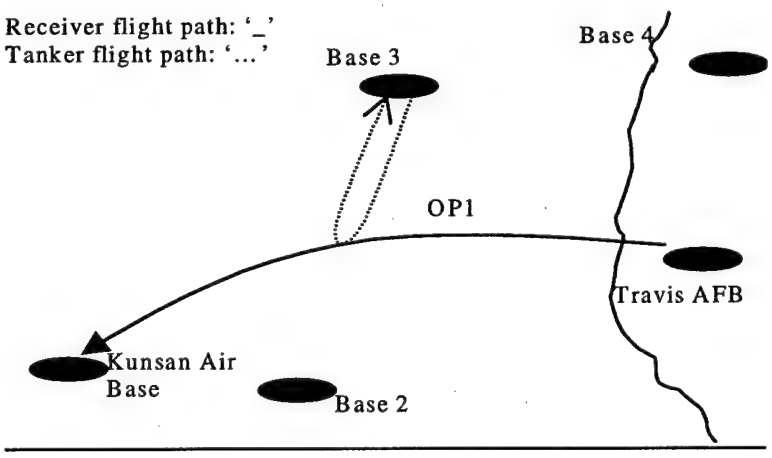

Figure 3: Tanker rendezvous with a C-17

Thus, another important consideration relating to the interaction of receivers and tankers is the development of a protocol governing the specification of desired refueling points. Specification of refueling points are driven by several variables including escort requirements, receiver fuel needs, and receiver fuel consumption rates.

Tanker Availability, Recycling, and Scheduling Precision. A problem closely related to the geographical situation is tanker availability. There may be tankers located at the origins or destinations of receiver aircraft or surrounding airbases close to the route of flight. We would like to find the best way to utilize all available tankers. Thus, in general we look for the closest available tankers to a desired refueling point or escort path.

Another issue related to the availability and positioning of tanker aircraft is the problem of tanker recycling. Tanker recycling reuses tankers to support multiple refueling points. Suppose two receiver groups require refueling to complete a mission, but only one tanker is available to refuel both refueling points. In this case, the tanker must support one of the two missions, then return to base for refueling before supporting the second flight. In this simple situation the best schedule is trivial since we typically support the more urgent mission first. However, more complex scheduling problems increase the complexity of analyzing tanker recycling.

An obvious corollary of the recycling issue is the question of scheduling precision. In our example deployment, the desired schedule is in terms of days. We may wish to schedule in terms of hours, even minutes. Thus, the question of tanker availability and recycling becomes very important. It requires knowing accurately the

amount of time after being assigned to a receiver that a tanker requires to complete the refueling operations, return to base, and become available for reassignment.

Crew Duty Limits. Another concern in scheduling deployment operations is limitations on flight duration due to crew duty limits. For the safety of flight crews, Air Force regulations limit the length of time a crew may fly without a rest period. Thus, similar to the issue of recycling tankers for repeated assignments, the availability of crews for assignment must be considered.

Take-off Fuel. The amount of aerial refueling required to complete a particular deployment mission can also be affected by the conditions of an aircraft's origin base, specifically runway length and weather conditions. These parameters effect the take-off weight of an aircraft in order to leave the ground. The primary method to decrease the weight of an aircraft for take-off is to decrease the amount of fuel onboard.

Aircraft Fuel Consumption

Russina, Ruthsatz, and Russ (1999) identify and model the factors that determine the fuel consumption rates of different aircraft.

True Air Speed. The true air speed of an aircraft directly influences the fuel consumption rate since as an aircraft travels at faster speeds, it will burn fuel more quickly.

Ground Speed. The amount of time necessary to traverse a particular distance is not completely determined by the air speed of the aircraft. The effects of wind conditions can increase or decrease the duration of a flight, causing changes in the fuel consumption. Furthermore, if an aircraft's airspeed adjusts to compensate for wind conditions, fuel consumption is also affected.

Altitude. Another important factor concerning modeling fuel consumption rates is the effect of flight altitude; i.e., fuel consumption rates are inversely proportional to the altitude of flight.

Gross Weight. The fuel consumption rates of aircraft are also affected by the total gross weight of an aircraft; i.e., fuel flow rates are directly proportional to the gross weight.

Refueling Maneuvers. Finally, receiver aircraft often decrease altitude from a more fuel efficient cruise altitude to a safe refueling altitude, and decrease speed to a safe refueling speed. These refueling maneuvers also affect overall fuel consumption.

The paper explaining the Quick Look Tool provides detailed equations for the above factors, along with examples for using them. The user interface involves a Microsoft Excel Workbook with ten worksheets. All calculations made by the Quick Look Tool are done in Excel macros written in Visual Basic for applications code. The current Quick Look Tool accounts for tanker availability on a day-by-day basis, but does not consider the tankers as located at separate bases.

2.2 Heuristic Approaches

The word "heuristic" is derived from the Greek word "heuriskein," meaning "to discover." In practice, heuristics algorithms provide near-optimal solutions to difficult problems in a timely and easy manner. According to Silver, Vidal, and Werra (1980) there are several possible reasons a heuristic optimization approach is desired. The problem formulation is such that an analytic (closed form) or iterative solution procedure is unknown. An exact analytic or iterative solution procedure may be computationally

limiting to use. The heuristic method may be simpler for the decision maker to understand. The method can be used as a learning tool when the problem's optimal solution is known. In addition, since models are necessarily inaccurate representations of the real conplex world problem, "optimal" solutions are only academic, and a heuristic solution suffices. A fast near-optimal solution makes more sense than a time-consuming exact solution to an inexact problem (Zanakis 1981).

"Good" heuristics have several properties or features in common: (i) they are *simple* which assists their understanding and acceptance; (ii) the *core* storage requirements are reasonable in size; (iii) the methods have *speed*; i.e., the computation times do not grow at polynomial or exponential rates as the problem size increase; (iv) the solutions obtained are *accurate*, and the degree of accuracy is determined by the problem structure or user; (v) the methods are *robust*, which means as the size and parameters of the problems change, the method still obtains good solutions, in reasonable times; (vi) *multiple starting points* that are not necessarily feasible are allowed; (vii) *multiple solutions* must be available by selecting proper input parameters, and this allows the user to select the solutions with the most accuracy or satisfaction; (viii) a good *stopping criteria* must be present, taking advantage of the search memory; (ix) the user must be able to *interact* with the method (Zanakis 1981). One of the better heuristic methods in use is tabu search (TS), first proposed by Glover (1986).

2.2.1 Tabu Search

TS is meant to cross boundaries of feasibility or local optimality, which were usually viewed as barriers, to derive a method for intelligent problem solving (Glover and

Laguna 1997). The Committee on the Next Decade of Operations Research (CONDOR 1988) considered tabu search to be "extremely promising" for the future treatment of practical applications.

Tabu search explores the solution space by systematically moving among solutions. At each iteration, a move is made to some best solution in the neighborhood of the current solution (not necessarily an improving solution). TS forbids, or makes tabu, solutions with certain attributes in order to prevent cycling and to direct the search to other regions of the solution space not yet explored. Short term and long term memory functions prevent solutions obtaining these attributes from occurring, primarily through measures of recency and frequency. The most common form of short term recency-based memory is the tabu list. A tabu list consists of solution attributes modified in the latest moves so that these recent solutions are not revisited. The size of the list determines the number of iterations a certain attribute is considered tabu and not allowed to influence the next move.

Tabu search requires the following items (Ben-Daya and Al-Fawzan 1998):

Initial solution. There are many methods to generate this initial solution, including random generation or greedy algorithms. This initial solution is not required to be a feasible solution.

Neighborhood generating mechanism. Each solution is associated with a neighborhood. This neighborhood contains all the solutions accessible within one move from the current solution. Moves are unique manipulations of certain attributes for a given solution.

Tabu list. A list containing attributes associated with recent moves; used to avoid solution cycling.

Stopping criteria. TS contains no method to determine when the optimal solution has been reached; thus, a method must be used to determine when the search procedure terminates. Common procedures include using solution tolerance and iteration counts.

Additional elements which prove useful to TS include:

> *Aspiration criterion.* If a tabu move satisfies a specific aspiration criterion, the move is considered among the other candidate solutions. A commonly used aspiration criterion accepts a tabu move if it produces a solution with a better objective function value than the best found so far.
>
> *Intensification scheme.* This method intensifies the search in a promising region of the space. This can be accomplished by locking in the choice of attributes frequently contained in good solutions.
>
> *Diversification scheme.* This method moves the solution to an area of the solution space not explored previously.

There are many methods of choosing a good balance between intensification and diversification schemes. These methods play an important role in tabu search. The Reactive Tabu Search (RTS) algorithm determines the value of a prohibition parameter in TS, so that a balance of exploration versus exploitation is obtained that is appropriate for the local characteristics of the task (Battiti 1996). The prohibition parameter in TS, usually referred to as the tabu tenure, determines the length of the tabu list which dictates the number of iterations certain attributes of the solution are considered "tabu."

The first main characteristic of RTS is the self-adjusting prohibition period. In RTS, the prohibition T, more commonly referred to as the tabu tenure, is determined through feedback mechanisms as the search processes. Initially, T is equal to one; it increases when there is a need to diversify; it decreases when this need vanishes. The need for diversification is signaled by the repetition of previously visited solutions.

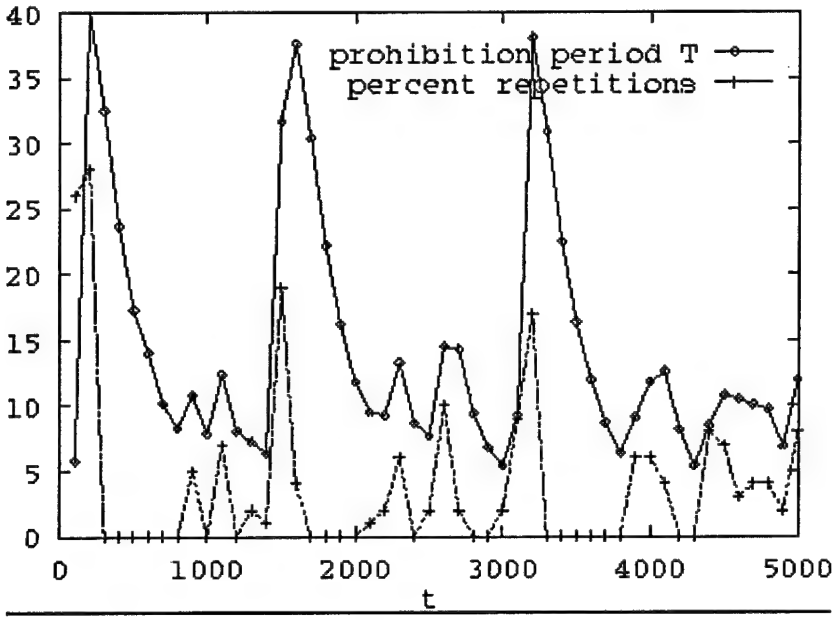

Figure 4: Tabu Tenure Behavior During RTS

Figure 4 shows the behavior of *T* during the TS applied to a Quadratic Assignment Problem (Battiti and Tecchiolli 1994). The value of *T* increases exponentially when repetitions are encountered and decreases gradually when repetitions disappear.

The escape mechanism represents the second main characteristic of RTS. The basic tabu method for prohibitions may not be sufficient to avoid long cycles. Even if "limited cycles" are avoided, the search may still be in a limited region of the search space. An additional escape mechanism making radical diversification steps is needed. The escape phase activates when too many configurations repeat too often in the search (Battiti and Tecchiolli 1994). A simple escape consists of random moves away from the current solution (possibly with a bias toward moves that translate the trajectory away from the current search region) (Battiti 1996).

The last main characteristic refers to fast algorithms and efficient memory storage for using the search history, typically through the use of hashing techniques. The main

purpose of a hashing function is to enable a tabu search algorithm to rapidly detect whenever it transitions to a previously visited solution. Woodruff and Zemel (1993) present four different hashing functions, each using a series of random integers in order to compute the hashing function, for use within a tabu search algorithm. According to the authors, there are three purposes for these hashing functions h. (i) The hashing functions should be as easy as possible to update and compute. This requires the structure of h to have characteristics similar to the structure of the neighborhood sets. (ii) The series of random integers should be in a range having reasonable storage requirement; i.e., an integer requiring two or four bytes. (iii) A low probability of collision is preferred. A collision occurs when two different solutions are encountered with the same hash function value.

The CPU time for h is approximately constant with respect to the number of iterations (Battiti 1996). Therefore, the overhead caused by the use of the history is negligible for problems requiring a non-trivial number of operations to evaluate the objective function value in the neighborhood.

2.3 Conclusion

Although the Quick Look Tool for Tanker Deployment designed by Russina, Ruthsatz, and Russ (1999) provides a means to schedule the tanker aircraft to receiver groups, it does not model multiple locations for these tankers. This research extends their code, increasing the tanker's capability to multiple origins. Due to the complexity of this problem, a heuristic is required to get solutions in a reasonable length of time. We use a modified tabu search (TS) method to solve this tanker assignment problem (TAP).

Chapter 3. Methodology

3.1 Solution Representation

In order to solve the TAP, we first determine how to represent the solution. There must be a tanker aircraft assigned to each refueling point, so the solution consists of these tankers assigned to the refueling points plus the time at which the tankers take-off from their origin base. In addition, the tanker take-off times directly determine the times at which the receiver groups must leave their base of origin. Thus, the receiver group take-off times are also part of a solution.

We must first generate the list of solution attributes, which are modified to form different solutions. Let **RG1** be the first receiver group request. We first determine if this receiver group requires refueling. If **RG1** must be refueled during its deployment, we calculate the location for the needed refueling(s). Let $\mathbf{RG1_1}$ be a decision variable (DV), representing the first refueling point for receiver group one. Next, we check each of the bases with tanker aircraft to determine if there exists a tanker capable of satisfying **RG1's** fuel requirements. If **RG1** consists of aircraft classified as "heavy," the amount of fuel that group requires is the amount of fuel needed to successfully continue the flight to its destination or to the next refueling point. On the other hand, if **RG1** contains aircraft classified as "fighter," then this group must have a tanker escort between refueling points. Although "fighter" groups only require tanker escort over open water, we assume for this model that they require tanker escort between each refueling point and their final leg of the mission. For this type of receiver group, a tanker must be capable of traveling to the refueling point, fulfill the receiver group's fuel requirements, escort the group to the next

refueling point, and finally return to its base of origin. For each tanker capable of satisfying the requirements at $\mathbf{RG1_1}$, we generate a number of alternatives for the decision variable $\mathbf{RG1_1}$. A separate alternative is generated for each tanker and for discrete take-off times, in one-hour increments. We generate this list of DV alternate values for all bases with tankers capable of satisfying the fuel requirements for $\mathbf{RG1_1}$.

Suppose that there exists no tanker with enough fuel to satisfy $\mathbf{RG1_1}$'s requirements. In this case, we attempt to assign two tankers to this refueling point, with each satisfying half of the off-load required of **RG1**. This process continues until the appropriate numbers of tankers are assigned to $\mathbf{RG1_1}$. For each tanker assigned to $\mathbf{RG1_1}$, a new DV is generated, with the same list of alternative choices for that DV. For example, if two tankers are needed for $\mathbf{RG1_1}$, then two DVs are generated, each having the same set of alternate tankers, since both DVs have the same fuel off-load.

If **RG1** requires more than one refueling point, we apply this same technique to each of the these extra refueling points, with each of them representing another DV; however, each tanker has only one take-off time, as opposed to the different distinct take-off times for the tankers assigned to a first refueling point. This take-off time is a continuous value that is based on the take-off time for the tanker assigned to the receiver group's first refueling point.

This method continues for each receiver group until all refueling points and numbers of tankers assigned to each are determined. At this point, the list of DV alternatives represents all options for assigning tankers to receiver groups for the final mission plan. Which of these alternatives to choose is now the focus of the search engine.

3.2 Greedy Construction Heuristic

To expedite the search process, we begin with an initial feasible mission plan. This plan is generated using a greedy construction heuristic. For each DV, representing a refueling point, we assign the closest tanker that has not yet been assigned to another DV. If the maximum number of tankers at the closest base is reached, we begin to repeat the tanker assignments at this base. This is when a tanker conflict may occur, in which the same tanker is scheduled for more than one refueling point at the same time. Increasing the number of tankers at each base eliminates this conflict, but also increases the computation time required during the search process. Therefore, we allow the initial solution to be infeasible for some deployment problems. When we begin with an infeasible solution, the first goal is to reach a feasible solution.

3.3 Solution Representation

A given solution to this deployment is represented by a mission plan. Each decision variable's (DV) value consists of a specific tanker assigned to a refueling point and the time that tanker is scheduled to depart its base of origin. The values of the DVs are used to calculate the time of departure (TOD) from the origin base and time of arrival (TOA) to the destination base for each receiver group. In order to determine the best DV values for the deployment, we must have a means of evaluating a mission plan.

3.4 Mission Evaluation

There are many specified goals for this problem. Our first goal is to have the receiver groups arrive at their destinations on time. This is represented by a hard constraint, resulting in an infeasible mission plan if any receiver groups are scheduled to

arrive past their RDD. The second goal is to minimize the total distance traveled by the tanker aircraft. Another goal is to use the fewest tankers possible. We evaluate a mission plan using a single value incorporating each goal:

Mission Evaluation = Distance Penalty – Reuse Bonus – Early Bonus + (Sync Penalty + Conflict Penalty + Tardiness Penalty + Early Penalty + Negative TOD Penalty) + Tanker Index

The TS search engine primarily uses the following three values during the search process.

Distance Penalty: For each decision variable, the distance penalty is the sum of the distances traveled by all tankers in the current mission. The solutions with smaller distance penalties are more attractive.

Reuse Bonus: Each time a tanker is reused in a mission, this value is increased.

Early Bonus: Although we are most concerned with having each receiver group arrive by its RDD, it is valuable to have them arrive earlier. Thus, this value represents the number of hours a receiver group arrives at its destination before its RDD.

The next set of values encourages feasibility throughout the search process.

Sync Penalty: If a receiver group requires more than one tanker at a refueling point along its path, these tankers must arrive at that location at the same time. A heavy penalty is applied when the tankers assigned to this multi-tanker refueling point are not scheduled to arrive at the same time.

Early Penalty: The deployment is scheduled to begin at time zero. Thus, a heavy penalty is applied when a receiver group's TOD is before time zero.

Negative TOD Penalty: In the case when a receiver groups TOD is before zero, which might happen when starting with an infeasible solution, this value directs the

search process into the feasible region. This term applys large negative values to the missions that correct this problem, thus making those missions attractive choices for the next mission plan.

Conflict Penalty: In order to ensure that a tanker does not refuel two receiver groups at the same time, we impose a conflict penalty. Each tanker is given a 3-hour turnaround period between missions. If any of the tankers in the solution break this 3-hour separation between assignments, a conflict occurs. When this happens, a large conflict penalty is added to the objective function.

When we use the greedy construction heuristic, the initial solution is generally feasible with no tanker conflicts. However, due to the number of tanker aircraft available at a base, the greedy construction heuristic can generate an initial solution with tanker conflicts. When this occurs, the search method must bring the solution back into the feasible region. To accomplish this, the large negative conflict penalty in the objective function encourages TS to generate moves to decrease the number of conflicts.

Tardiness Penalty: In order to ensure that each receiver group arrives before its RDD, a large penalty is applied to the objective function when a receiver group's TOD is tardy.

Tanker Index: Tankers are indexed at each base. Indexing encourages using the lower indexed tankers. We use this value for cosmetic reason only, since all tankers are of the same type.

3.5 Tabu Implementation

The tabu search employed for the TAP explores the solution space by swapping tankers assigned to a refueling point. A *move* is defined as removing an existing tanker assigned to a refueling point and replacing it with another tanker within range to satisfy the needs of the receiver group at that refueling point.

To illustrate, consider the first refueling point in the mission. The current tanker assigned to this refueling point is removed. Note that the decision variable representing this tanker assignment also has a take-off time corresponding to the tanker base and number. A decision variable representing a different tanker assigned to this refueling point is included in the solution. This new (temporary) mission plan, or neighbor, is evaluated. This evaluation is made for each tanker/take-off time within range of the first refueling point. This process is repeated for each refueling point in the deployment mission. At the end of this process, there is a mission evaluation for each decision variable alternative, representing all possible changes to the mission plan making only one move. This represents the candidate list for the current iteration.

TS now chooses the best mission evaluation from the candidate list. Once this move is made, TS puts this refueling point on the tabu list. TS will not change the tanker assigned to a refueling point on this list unless its mission evaluation is better than any mission thus far in the search process. This is known as the aspiration criterion. A refueling point will remain on the tabu list for a set number of iterations. This number is referred to as the tabu tenure. Changing this tenure affects the behavior of the search process. TS relies on this tenure to regulate its short-term memory characteristic.

Humans tend to have a short-term memory capacity of 7, and empirical TS results find this a reasonable value to start with as a tenure length (Glover 1990).

3.5.1 Tabu Search Methods

The TAP tool allows the user to modify three parameters regarding the tabu search process. In addition to changing the tabu tenure, the user may select to change two other parameters. The first parameter determines the candidate list size during the search process. Due to the complexity of TAP, computation time greatly increases as the number of DVs increase. Therefore, the *skip number* directs TS to only consider portions of the candidate list. The skip number equates to the number of portions of the candidate list considered. With a skip number of one, there is no reduction in the candidate list size. TS considers every DV alternative at each iteration. If the skip number is two, TS considers every other DV alternative. For example, if the DV alternatives were numbered, then the first candidate list consists of the mission evaluations for the odd numbered alternatives. For the next iteration, only the even numbered alternatives are evaluated. The candidate list is split in a similar manner for skip numbers greater than two.

The second parameter effects the size of the tabu restriction. If the user selects the *large* restriction, TS will put the refueling point DV on the tabu list. Thus, TS restricts the changing of this DV unless it satisfies the aspiration criterion. On the other hand, selecting the *small* restriction puts the refueling point DV and the base assigned to that DV on the tabu list. In this case, TS may assign a tanker from a different base to this DV, but assigning a tanker from the base already assigned to the DV is restricted.

By changing these two parameters we form different TS methods. If we only allow the skip number to be one or two, representing the *full* and *half* size candidate list respectively, we form four specific TS methods. We call these four methods *Full/Large*, *Full/Small*, *Half/Large*, and *Half/Small*.

3.6 TAP Tool Implementation

Once AMC inputs the data for a deployment scenario, the TAP tool uses three phases to arrive at an output consisting of the initial, final, and best mission plans (see Figure 5). The three phases consist of DV alternative generation, initial solution generation, and tabu search. Detailed flow charts for each of these three phases are depicted in Figures 6-8, respectively.

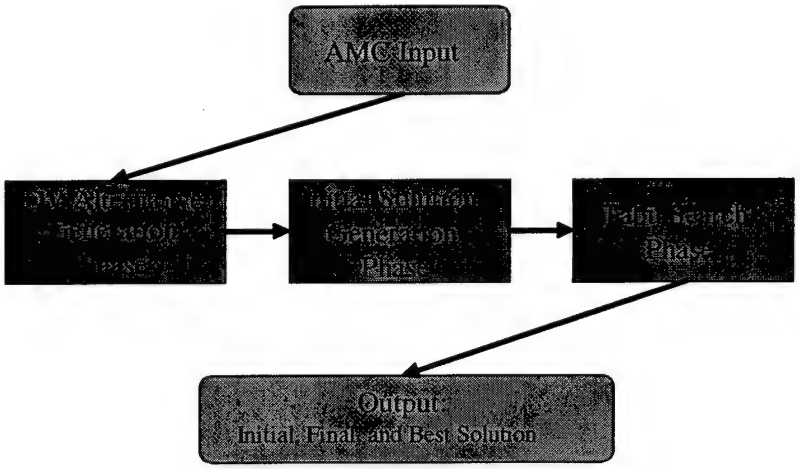

Figure 5: TAP Tool Flow Chart

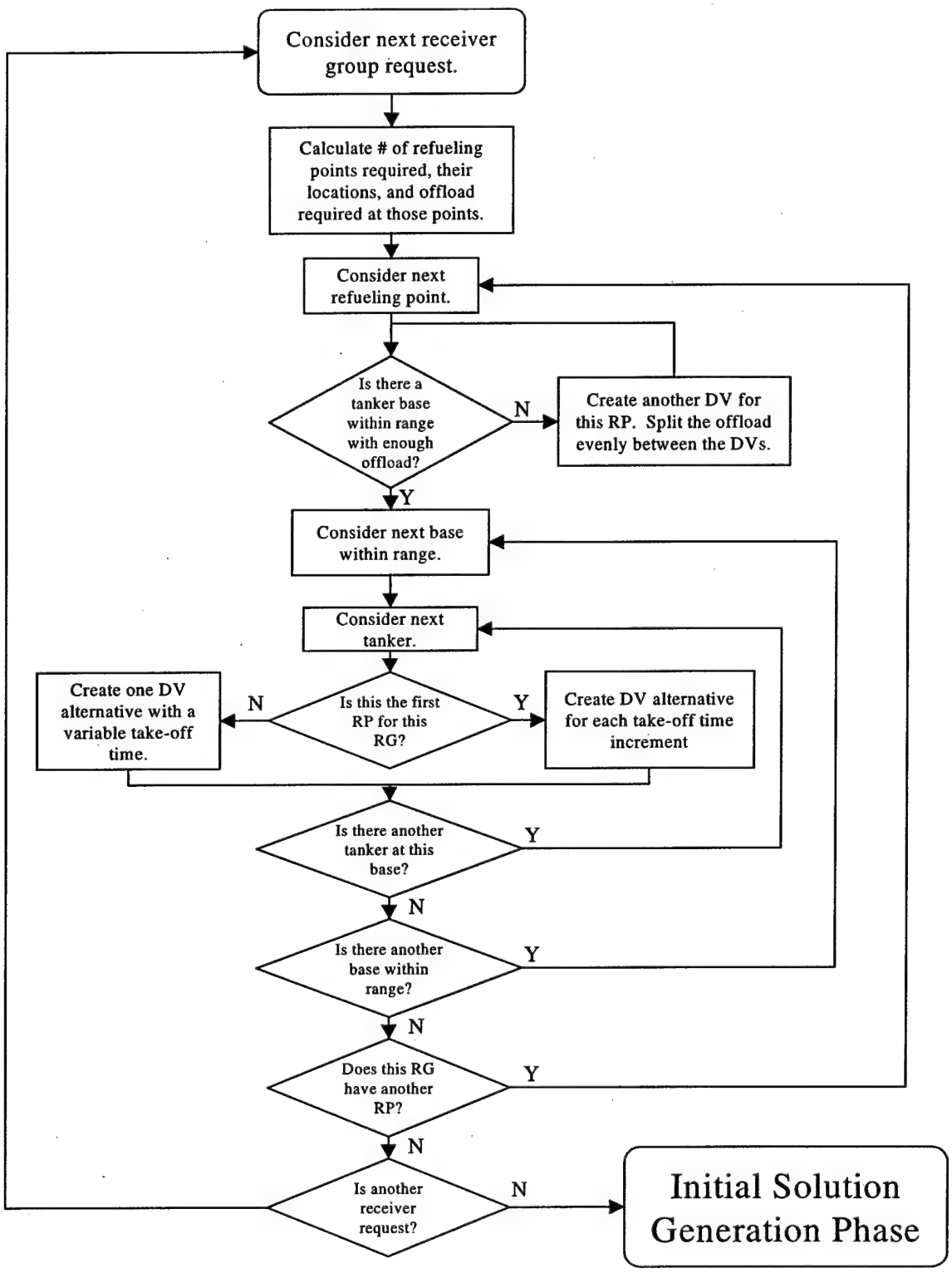

Figure 6: DV Alternative Generation Phase

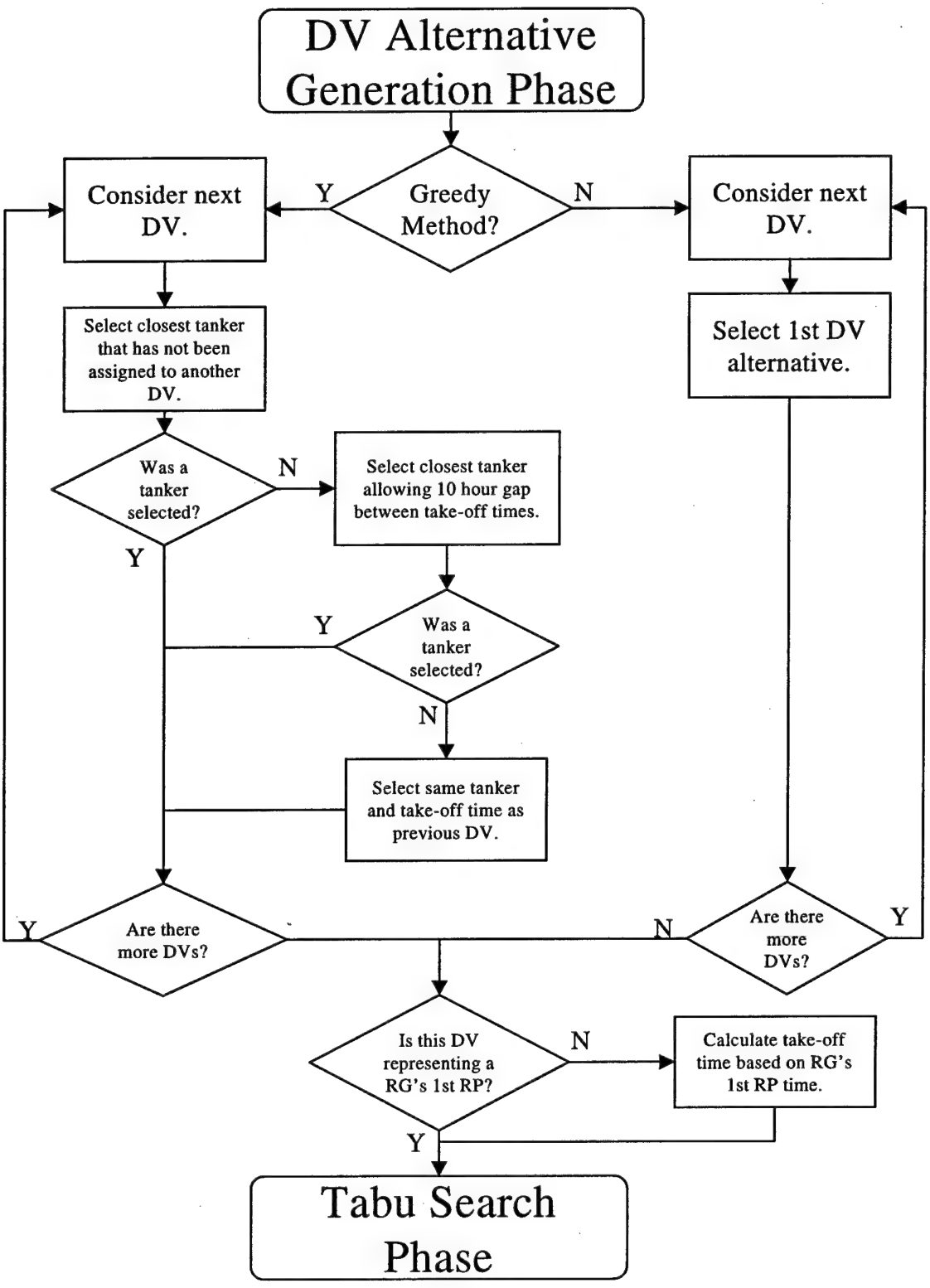

Figure 7: Initial Solution Generation Phase

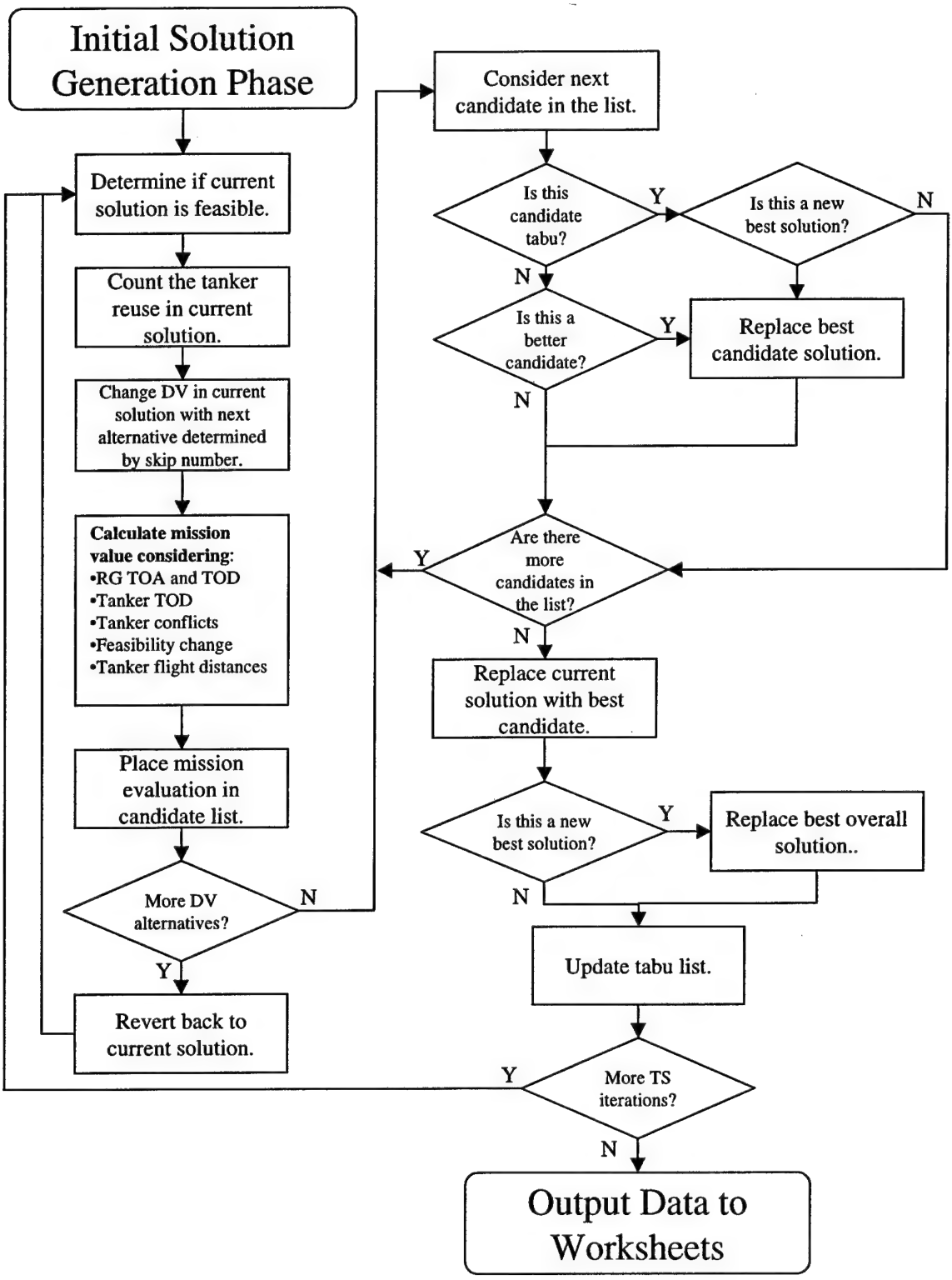

Figure 8: Tabu Search Phase

3.7 Measurement of Results

We measure our results through a comparison with the solution generated by the current tanker-planning tool, CMARPS. AMC has set a conservative goal to achieve results within 20% of CMARPS' outputs; however, the goal of this research is to have computation times much less than CMARPS' and produce solutions that have the same quality as CMARPS' solutions. In the next chapter, we apply TS to two sample deployments. We also compare the CMARPS to the TAP Tool for a simple mission plan.

Chapter 4. Results and Conclusions

An Excel-based tool was developed to input a number of receiver group requests and output a mission plan consisting of tanker assignments to refueling points. Two sample deployments were provided by AMC for testing this new tool. In addition, AMC provided bed-down locations of KC-135s around the globe.

Since this tool considers all tankers within range of refueling points, it is necessary to decrease the actual numbers and locations of tankers in order to increase the computational efficiency of the tool. For example, there are several Air National Guard and Air Force Reserve bases in the U.S. that contain tanker aircraft. Many of these bases are within range of the early refueling points of a deployment. This large number of available tankers greatly increases the number of decision variable alternatives during the search process. Therefore, we first run the program with tankers located at active airbases. If the tool returns a solution with no tanker assigned to a refueling location, we place tankers at a base originally in the list of bed-down locations so that there is a tanker capable of satisfying the receiver group's requirements at that refueling point. Due to the time required to run this tool, we make the first run using only one iteration of TS. This provides us with a list of all bases with tankers within range of the refueling points. Once there is a tanker within range of each refueling point, we make a full TS run. The number of TS iterations performed in the full run is determined by the complexity of the deployment and the user's time constraint.

4.1 Southeast Asia Deployment

The first deployment we test involves receiver groups departing the continental U.S. and arriving in Southeast Asia. Table 2 provides a list of the 11 receiver groups shown in Figure 9.

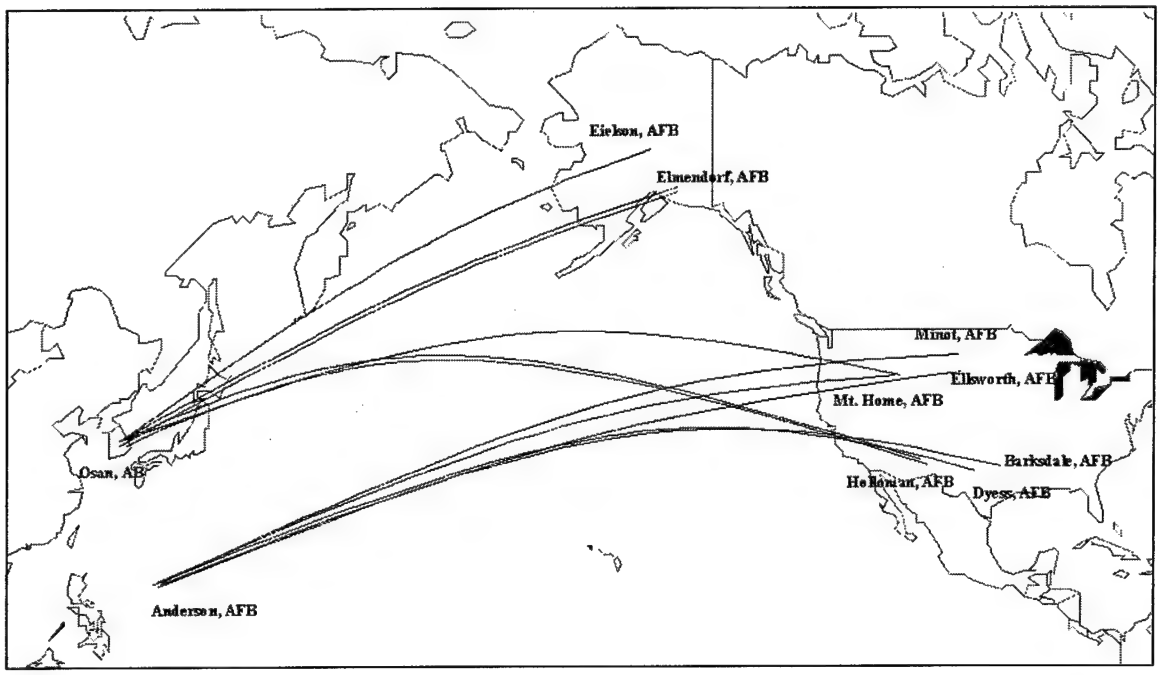

Figure 9: Southeast Asia Deployment

The tanker bases activated for this deployment include McConnell, Mountain Home, Grand Forks, Fairchild, Kadena, and Eielson, with 15 KC-135 tankers located at each base. Tankers located at these bases are capable of satisfying all the receiver groups' fuel requirements during the deployment. None of the receiver groups have waypoints for this deployment. A waypoint is a location the receiver group must first

reach before heading towards their destination base. Our code allows the user to input one waypoint for each receiver group.

Table 2: Receiver Groups for Southeast Asia Deployment

RG #	Aircraft Type	# of Aircraft	Origin	Destination	ALD	RDD
1	F117	2	Holloman	Osan	1	5
2	F15	6	Mountain Home	Osan	1	5
3	F15	6	Elmendorf	Osan	1	5
4	F16	6	Eielson	Osan	1	5
5	A/OA10	6	Eielson	Osan	1	5
6	B1	1	Mountain Home	Andersen	1	5
7	B1	1	Ellsworth	Andersen	1	5
8	B1	1	Dyess	Andersen	1	5
9	B52	1	Barksdale	Andersen	1	5
10	B52	1	Minot	Andersen	1	5
11	F117	2	Holloman	Osan	1	5

We apply the *Half/Small* TS method with a tabu tenure of 7. Table 3 shows the resulting initial mission plan generated by the greedy method. Table 4 displays the TOD and TOA for each receiver group. The TOD and TOA values for tankers and receiver groups are represented in hours after the deployment begins.

Table 3: Initial Mission Plan for Deployment 1

RG #	Refueling Point #	Tanker Base	Tanker #	Tanker TOD (hours)
1	1	FAIRCHILD AFB	1	6.0
1	2	EIELSON AFB	1	7.7
1	3	EIELSON AFB	2	9.4
1	4	KADENA AB	1	10.4
1	5	KADENA AB	2	15.5
2	1	EIELSON AFB	3	6.0
2	1	EIELSON AFB	4	6.0
2	2	KADENA AB	3	5.7
2	2	KADENA AB	4	5.7
2	2	KADENA AB	5	5.7
2	3	KADENA AB	6	13.3
3	1	KADENA AB	7	1.0
3	1	KADENA AB	8	1.0
3	1	KADENA AB	9	1.0
3	2	KADENA AB	10	8.6
4	1	EIELSON AFB	5	1.0
4	1	EIELSON AFB	6	1.0
4	2	KADENA AB	11	3.0
5	1	KADENA AB	12	1.0
5	1	KADENA AB	13	1.0
5	2	KADENA AB	14	10.5
6	1	KADENA AB	15	11.0
7	1	KADENA AB	1	21.0
8	1	KADENA AB	1	31.0
9	1	EIELSON AFB	7	11.0
9	1	EIELSON AFB	8	11.0
10	1	EIELSON AFB	9	6.0
11	1	FAIRCHILD AFB	2	6.0
11	2	EIELSON AFB	10	7.7
11	3	EIELSON AFB	11	9.4
11	4	KADENA AB	12	10.4
11	5	KADENA AB	11	15.5

This initial mission plan consists of 28 tankers. Each receiver group arrives at its destination before their RDD. However, this mission plan is not feasible since there is a conflict between the two highlighted tankers. Tankers 11 and 12 out of Kadena are both

scheduled to depart before they have time to accomplish their first assignments and take a 3-hour maintenance period.

Table 4: Receiver Group Initial TOD and TOD for Deployment 1

Receiver Group #	TOD (hours)	TOA (hours)
1	4.5	17.3
2	4.1	15.2
3	3.6	11.0
4	1.1	8.5
5	1.1	12.3
6	4.1	16.2
7	16.2	29.5
8	25.8	40.2
9	4.6	19.6
10	0.6	13.8
11	4.5	17.3

The first course of action for the search process is to change the DVs values to make the mission plan feasible. This is accomplished during two iterations of TS. After 100 iterations, we arrive at the best mission plan found which is presented in Table 5 and Table 6. This plan consists of 13 tankers.

Table 5: Best Mission Plan Evaluation for Deployment 1

RG #	Refueling Point #	Tanker Base	Tanker #	Tanker TOD (hours)
1	1	FAIRCHILD AFB	1	6.0
1	2	EIELSON AFB	1	7.7
1	3	EIELSON AFB	2	9.4
1	4	KADENA AB	1	10.4
1	5	KADENA AB	11	15.5
2	1	EIELSON AFB	5	51.0
2	1	EIELSON AFB	4	51.0
2	2	KADENA AB	3	50.7
2	2	KADENA AB	4	50.7
2	2	KADENA AB	13	50.7
2	3	KADENA AB	11	58.3
3	1	KADENA AB	3	1.0
3	1	KADENA AB	8	1.0
3	1	KADENA AB	13	1.0
3	2	KADENA AB	10	8.6
4	1	EIELSON AFB	5	1.0
4	1	EIELSON AFB	4	1.0
4	2	KADENA AB	11	3.0
5	1	KADENA AB	3	31.0
5	1	KADENA AB	13	31.0
5	2	KADENA AB	11	40.5
6	1	KADENA AB	11	26.0
7	1	KADENA AB	3	16.0
8	1	KADENA AB	13	66.0
9	1	EIELSON AFB	5	36.0
9	1	EIELSON AFB	4	36.0
10	1	EIELSON AFB	4	16.0
11	1	FAIRCHILD AFB	1	61.0
11	2	EIELSON AFB	10	62.7
11	3	EIELSON AFB	4	64.4
11	4	KADENA AB	11	65.4
11	5	KADENA AB	3	70.5

Table 6: Receiver Group Final TOD and TOA for Deployment 1

Receiver Group #	TOD (hours)	TOA (hours)
1	4.5	17.3
2	49.1	60.2
3	3.6	11.0
4	1.1	8.5
5	31.1	42.3
6	19.1	31.2
7	11.2	24.5
8	60.8	75.2
9	29.6	44.6
10	10.6	23.8
11	59.5	72.3

The computation time for this run is approximately 20 minutes[1]. For the initial solution, the TAP Tool attempts to have each receiver group arrive as early as possible (see Figure 10). In order obtain a feasible solution and accomidate tanker reuse, TS distributes the receiver group flight times throughout the allowable deployment period (see Figure 11).

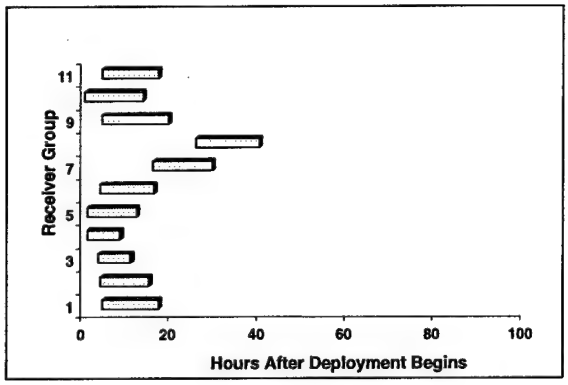

Figure 10: Initial RG flight times (Dep 1)

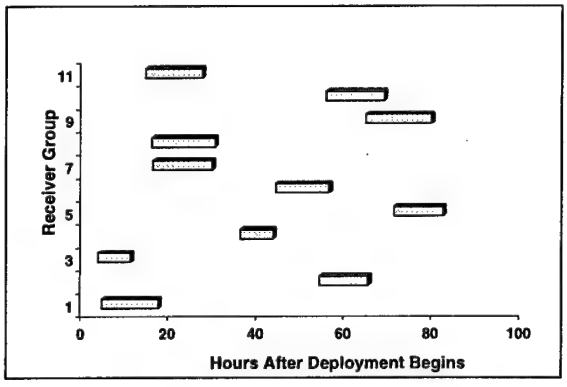

Figure 11: Best RG flight times (Dep 1)

We see in Figure 12 the mission evaluations during the 100 iterations of TS using the *Half/Small* method with tenure of 7.

[1] Intel Pentium II 350 Mhz, 64 Meg RAM

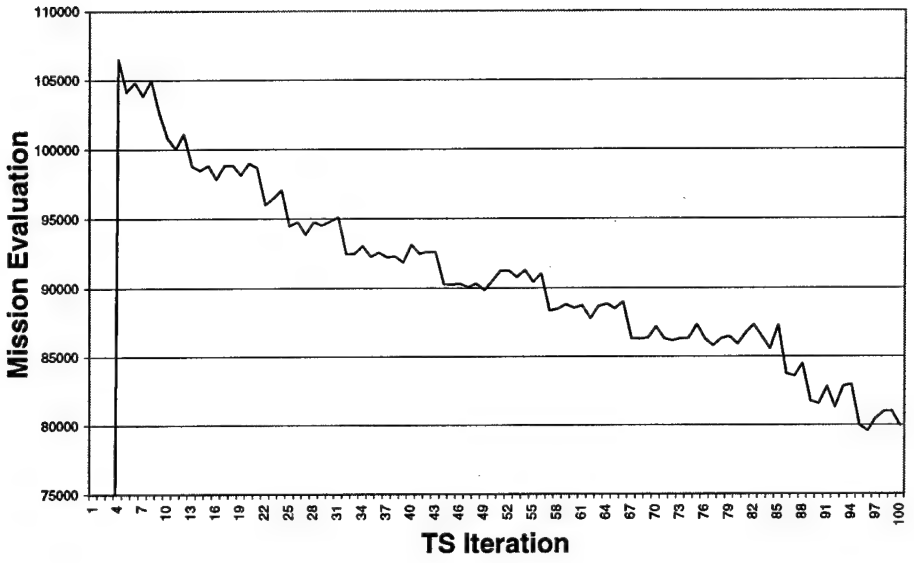

Figure 12: Southeast Asia Deployment TS Results

4.1.1 Tabu Tenure Comparison for Southeast Deployment

To examine the effects of the tabu tenure on this deployment, we perform six separate runs using odd tabu tenures 1 through 11.

Figure 13 shows the mission evaluations for these runs during the search process. For tenures 1 and 3, the search process is trapped in a local optimal region of the solution space around iteration 30. With a tenure of 5, the latter local optimal region is surpassed, but the search process eventually is trapped in another one around iteration 70. The final three tenure values (7,9,and 11) continue to navigate the solution space throughout the 100 iterations without being stuck within a local optimal region. The tenure values did not impact the computational times for TS.

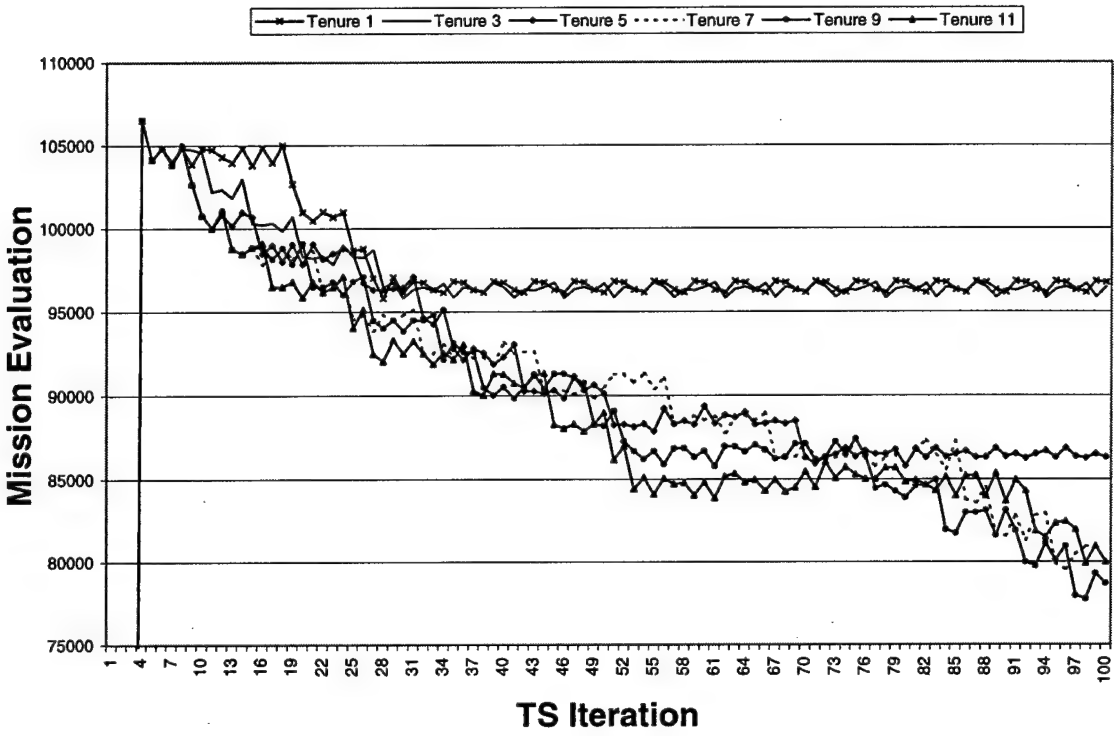

Figure 13: Tenure Comparisons for Deployment 1

Table 7 shows the statistics for the best missions resulting from the various tenure values. Tenures 7, 9, and 11 take strong advantage of tanker reuse in the deployment.

Table 7: Tabu Tenure Comparisons

Tabu Tenure	# of Tankers Used	Total Tanker Distance	Latest Receiver Group TOA (hours)
1	21	120461	78.5
3	21	120461	75.2
5	16	120461	70.1
7	13	120461	82.3
9	12	120461	73.5
11	13	120461	82.3

4.1.2 Tabu Search Method Comparison for Southeast Deployment

We next compare the four search methods, again applying 100 iterations for each TS method. We use a tabu tenure of length 7 to maintain consistency. Figure 14 shows the mission evaluation during the 100 iterations.

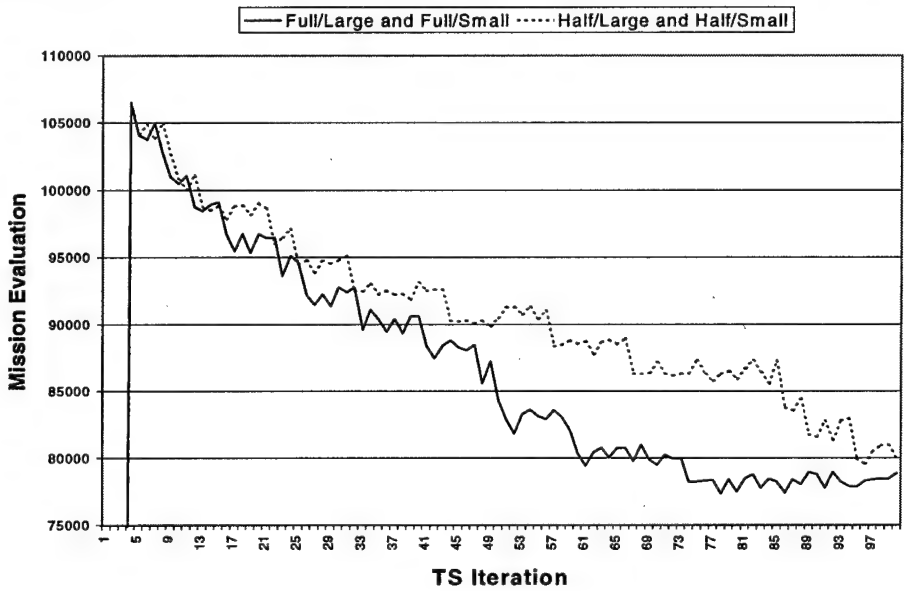

Figure 14: TS Method Comparison for Deployment 1

For this deployment, the size of the tabu restriction (large or small) had no effect on the search results. On the other hand, the candidate list size made a large difference. The resulting best mission evaluations are relatively close, but the computation time changed dramatically. Evaluating the full candidate list at each iteration resulted in 100 iterations of TS computed in 40 minutes. However, TS requires only 20 minutes to compute the 100 iterations when only half the candidate list is considered at each iteration.

4.2 Middle East Deployment

The second deployment we test involves receiver groups departing the continental U.S. and arriving in the Middle East region. Table 8 provides a list of the 9 receiver groups for the deployment shown in Figure 15.

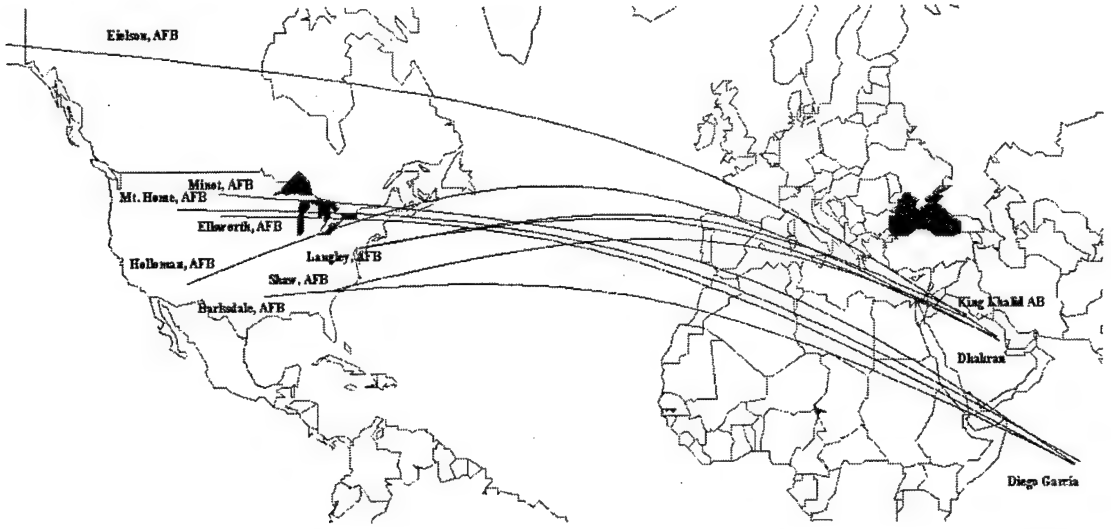

Figure 15: Middle East Deployment

Table 8: Receiver Groups for Middle East Deployment

RG #	Aircraft Type	# of Aircraft	Origin	Destination	ALD	RDD
1	F117	2	Holloman	Dhahran	1	5
2	A/OA10	6	Eielson	Dhahran	1	5
3	F15	6	Langley	Dhahran	1	5
4	F16	6	Shaw	King Khalid	1	5
5	F15	6	Langley	King Khalid	1	5
6	B1	1	Mountain Home	Diego Garcia	1	5
7	B1	1	Ellsworth	Diego Garcia	1	5
8	B52	1	Dyess	Diego Garcia	1	5
9	B52	1	Minot	Diego Garcia	1	5

The tanker bases activated for this deployment include Mildenhall, Bangor, Seymour-Johnson, and Eielson with 30 KC-135 tankers located at Mildenhall and 10 KC-135 tankers located at the other three bases. Currently, Mildenhall is the only airbase in the European region for the tanker bed-down. Because of this, many of the refueling points for this deployment must be satisfied by tankers located at Mildenhall. That is why we must station more KC-135s at that location. The user may choose to station some tankers at other bases in the region to share the workload with the ones stationed at Mildenhall. None of the receiver groups have waypoints for this deployment. Table 9 shows the resulting initial mission plan generated by the greedy method. Table 10 displays the TOD and TOA for each receiver group.

Table 9: Initial Mission Plan for Deployment 2

RG #	Refueling Point #	Tanker Base	Tanker #	Tanker TOD (hours)
1	1	BANGOR IAP	1	1.0
1	2	BANGOR IAP	2	3.2
1	3	MILDENHALL	1	5.6
1	4	MILDENHALL	2	9.5
1	5	MILDENHALL	3	10.6
1	6	MILDENHALL	4	10.8
2	1	MILDENHALL	5	1.0
2	1	MILDENHALL	6	1.0
2	1	MILDENHALL	7	1.0
2	1	MILDENHALL	8	1.0
2	1	MILDENHALL	9	1.0
2	1	MILDENHALL	10	1.0
2	2	MILDENHALL	11	8.0
2	2	MILDENHALL	12	8.0
2	2	MILDENHALL	13	8.0
2	2	MILDENHALL	14	8.0
2	2	MILDENHALL	15	8.0
2	2	MILDENHALL	16	8.0
2	3	MILDENHALL	17	12.4
3	1	MILDENHALL	18	1.0
3	1	MILDENHALL	19	1.0
3	2	MILDENHALL	20	8.8
3	2	MILDENHALL	21	8.8
3	3	MILDENHALL	22	9.2
3	3	MILDENHALL	23	9.2
3	3	MILDENHALL	24	9.2
4	1	BANGOR IAP	3	1.0
4	2	MILDENHALL	25	2.6
4	3	MILDENHALL	26	7.0
4	4	MILDENHALL	27	7.4
4	4	MILDENHALL	28	7.4
4	4	MILDENHALL	29	7.4
5	1	MILDENHALL	30	1.0
5	1	MILDENHALL	30	1.0
5	2	MILDENHALL	29	8.0
5	2	MILDENHALL	29	8.0
5	3	MILDENHALL	28	9.1
5	3	MILDENHALL	28	9.1
5	3	MILDENHALL	28	9.1
5	3	MILDENHALL	28	9.1
6	1	EIELSON AFB	1	11.0
7	1	BANGOR IAP	4	11.0
7	1	BANGOR IAP	5	11.0
8	1	BANGOR IAP	6	11.0
8	1	BANGOR IAP	7	11.0
9	1	BANGOR IAP	8	11.0

Table 10: Receiver Group Initial TOD and TOD for Deployment 2

Receiver Group #	TOD (hours)	TOA (hours)
1	1.5	17.2
2	1.5	19.4
3	2.1	15.5
4	0.0	14.1
5	2.2	15.8
6	1.3	21.3
7	3.6	23.6
8	2.1	23.2
9	0.4	19.8

This initial mission plan uses 39 tankers. Each receiver group arrives at its destination before their RDD. However, this mission plan is not feasible since there is a conflict between tankers highlighted. Several of the tankers out of Mildenhall have schedules that overlap. Again, the search process is designed to first change the DVs' values to make the mission plan feasible. This is accomplished using seven iterations of TS. Using the *Half/Small* TS method with tenure of 7, the best mission plan we find after 100 iterations is shown in Table 10 and Table 11. This mission plan uses 26 tankers instead of the 39 assigned in the initial plan.

Table 11: Best Mission Plan Evaluation for Deployment 2

RG #	Refueling Point #	Tanker Base	Tanker #	Tanker TOD (hours)
1	1	BANGOR IAP	1	1.0
1	2	BANGOR IAP	2	3.2
1	3	MILDENHALL	1	5.6
1	4	MILDENHALL	2	9.5
1	5	MILDENHALL	3	10.6
1	6	MILDENHALL	4	10.8
2	1	MILDENHALL	6	1.0
2	1	MILDENHALL	20	1.0
2	1	MILDENHALL	26	1.0
2	1	MILDENHALL	24	1.0
2	1	MILDENHALL	22	1.0
2	1	MILDENHALL	10	1.0
2	2	MILDENHALL	11	8.0
2	2	MILDENHALL	12	8.0
2	2	MILDENHALL	13	8.0
2	2	MILDENHALL	14	8.0
2	2	MILDENHALL	15	8.0
2	2	MILDENHALL	16	8.0
2	3	MILDENHALL	17	12.4
3	1	MILDENHALL	26	41.0
3	1	MILDENHALL	6	41.0
3	2	MILDENHALL	20	48.8
3	2	MILDENHALL	10	48.8
3	3	MILDENHALL	22	49.2
3	3	MILDENHALL	23	49.2
3	3	MILDENHALL	24	49.2
4	1	BANGOR IAP	1	56.0
4	2	MILDENHALL	25	57.6
4	3	MILDENHALL	26	62.0
4	4	MILDENHALL	20	62.4
4	4	MILDENHALL	6	62.4
4	4	MILDENHALL	10	62.4
5	1	MILDENHALL	22	21.0
5	1	MILDENHALL	24	21.0
5	2	MILDENHALL	26	28.0
5	2	MILDENHALL	6	28.0
5	3	MILDENHALL	28	29.1
5	3	MILDENHALL	8	29.1
5	3	MILDENHALL	4	29.1
5	3	MILDENHALL	20	29.1
6	1	EIELSON AFB	1	11.0
7	1	BANGOR IAP	1	36.0
7	1	BANGOR IAP	5	36.0
8	1	BANGOR IAP	1	16.0
8	1	BANGOR IAP	7	16.0
9	1	BANGOR IAP	1	26.0

Table 12: Receiver Group Final TOD and TOA for Deployment 2

Receiver Group #	TOD (hours)	TOA (hours)
1	1.5	17.2
2	1.5	19.4
3	42.1	55.5
4	55.0	69.1
5	22.2	35.8
6	1.3	21.3
7	28.6	48.6
8	7.1	28.2
9	15.4	34.8

The greedy construction method selects early tanker take-off times putting the receiver groups flight times in the beginning of the deployment period (see Figure 16). The initial solution for the Middle East deployment has 10 tanker conflicts. TS encourages a feasible solution and tanker reuse by distributing the receiver group flight times across the deployment period (see Figure 17)

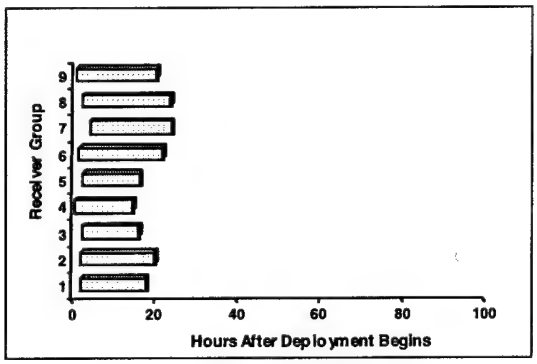

Figure 16: Initial RG flight times (Dep 2)

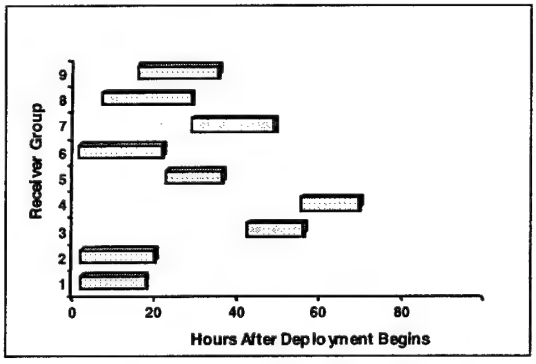

Figure 17: Best RG flight times (Dep 2)

The computation time for this run is approximately 1 hour and 30 minutes[2]. Figure 18 displays the mission evaluations during the 100 iterations.

[2] Intel Pentium II 350 Mhz, 64 Meg RAM

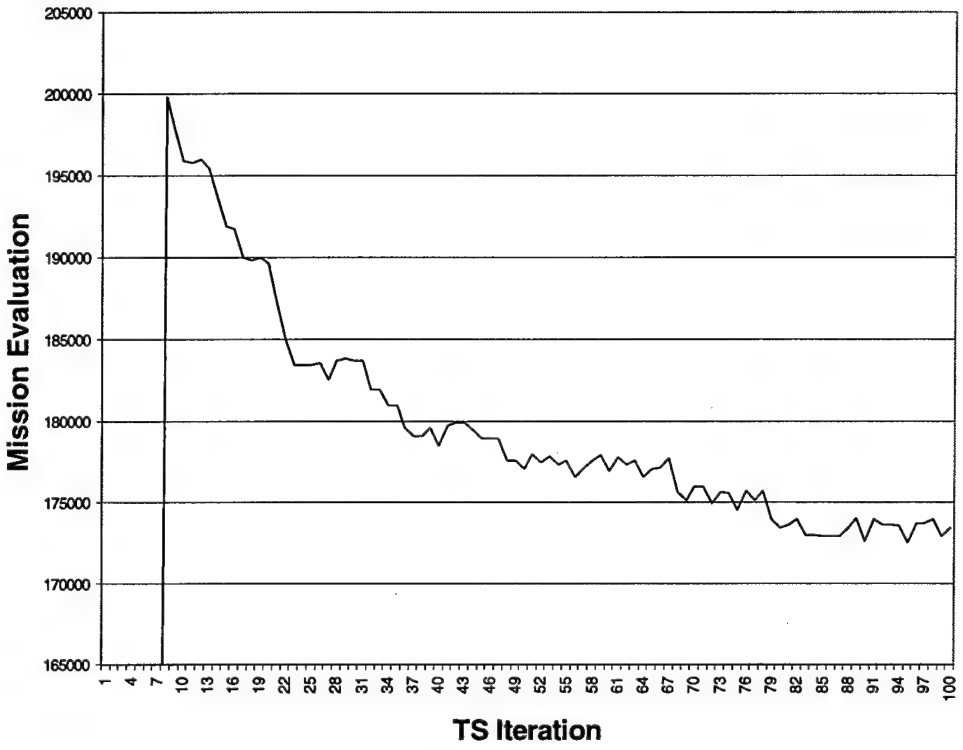

Figure 18: Middle East Deployment TS Results

4.2.1 Tabu Tenure Comparison for Middle East Deployment

We again compare the results based on 100 TS iterations using different tabu tenure lengths. We test with the odd tenure lengths from 1 to 11. Figure 19 shows the mission evaluations during these six runs.

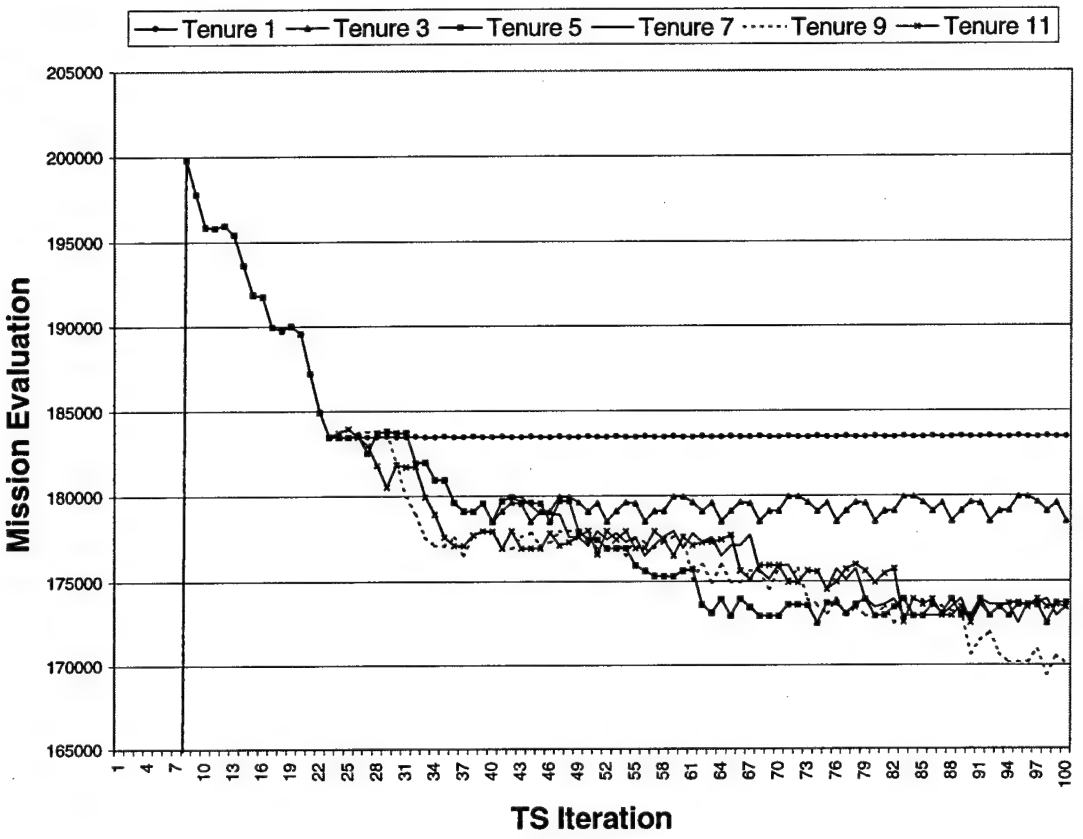

Figure 19: Tenure Comparisons for Deployment 2

The TS with tenure 1 quickly reaches a local optimum region and is trapped for the remainder of the iterations. Tenure 3 escapes this local optimum region, but is trapped within another region around iteration 40. Tenures 5, 7, and 11 continue to find better solutions, but appear trapped in a local optimum region during the last set of iterations. Tenure 9 has clearly escaped this local optimum region, and has found the overall best mission evaluation for the different TS tenure lengths. Table 13 shows that although the total tanker distance and latest receiver group TOA are worse with tenure 9, there are 2 fewer tankers used in the mission than with tenures 5, 7, and 11. Since tanker

reuse is weighted more than tanker distance and receiver group TOA in the mission evaluation, this mission is still considered the better one.

Table 13: Tabu Tenure Comparisons

Tabu Tenure	Number of Tankers Used	Total Tanker Distance	Latest Receiver Group TOA (hours)
1	31	215204	69.1
3	29	215204	69.1
5	26	215204	69.1
7	26	215204	69.1
9	24	216164	79.1
11	26	215204	69.1

4.2.2 Tabu Search Method Comparison for Middle East Deployment

We next compare the four search methods, again applying 100 iterations for each TS method. We apply a tenure of length 7 to maintain consistency. Figure 20 shows the mission evaluation during the search process. For this comparison, the TS runs for *Full/Large* and *Full/Small* use 50 iterations, while the other two use 100 iterations. The computation time required for each of these runs is 1 hour 26 minutes.

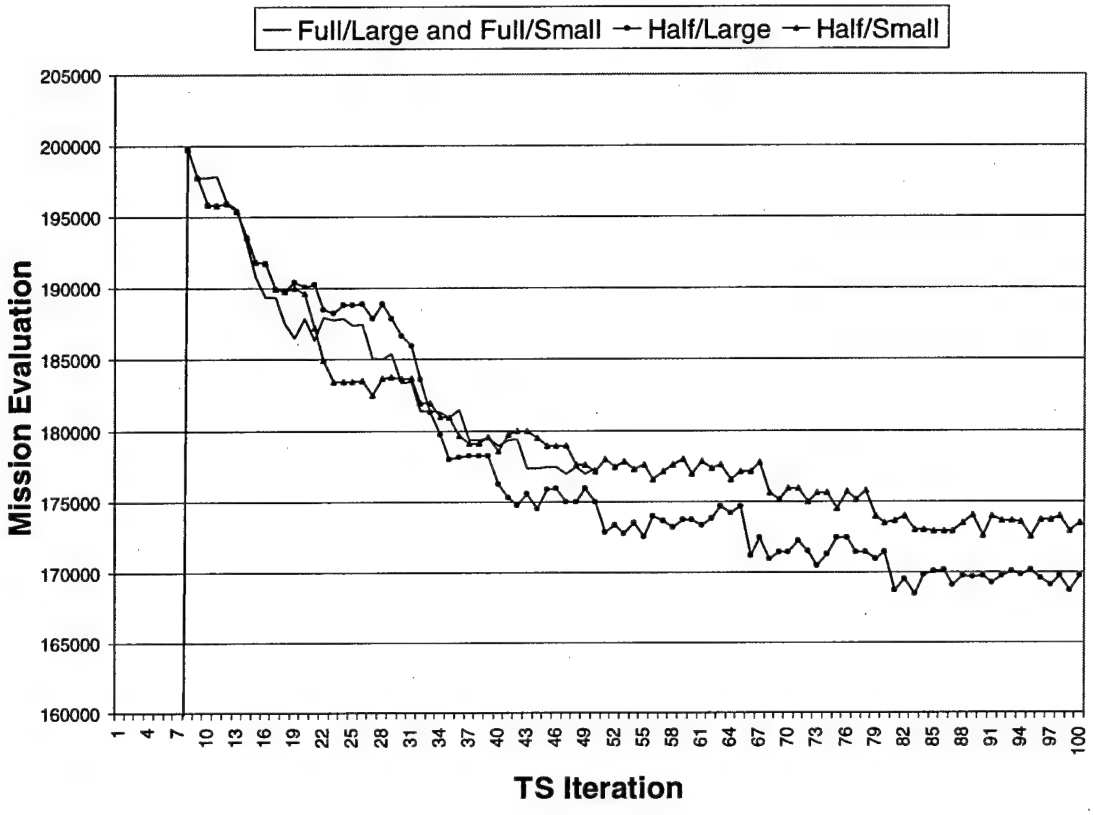

Figure 20: TS Method Comparison for Deployment 2

Although the size of the tabu restriction has no effect when TS considers the full candidate list, there is a difference when TS only considers half the candidate list for each iteration. Using a larger restriction for the *half* TS method found a better solution during the 100 iterations. This could be due to the fact that a larger restriction puts a larger number of DV alternatives on the tabu list. Essentially, this might have a similar effect to using a larger tabu tenure, since a larger tenure length also results in a larger amount of DV alternatives on the tabu list. Because more DV alternatives are considered tabu, the search process is forced to escape local regions. Providing TS with more DV alternatives to choose at each iteration allows the search process to explore local regions more

thoroughly, though possibly increasing the number of iterations needed to make critical moves to reach better solutions.

For this deployment, the TS methods considering half the candidate list at each iteration both reach better mission evaluations in the same amount of computation time. Furthermore, TS method *Half/Large* reaches a better solution in the first 50 iterations. This suggests that for similar deployments, runs should be made with this tool using a skip number of 2 and a large tabu restriction. Not only will the computation time be cut in half, but depending on the deployment scenario, this may be the TS method which produces the best mission evaluation.

4.3 CMARPS vs. TAP Tool

In order to compare the effort required by CMARPS to that of the TAP tool, we use two simple mission plans. The first mission plan involves a single B-52 receiver group scheduled to fly from the U.S. to Souda. We look at both one-way and round-trip flights. AMC is interested in the comparison of computation time and total offload required by the B-52 for this mission. Table 14 displays the comparison of the two tools.

Table 14: Tool comparison for B-52 mission

	One-way (KBAD→LGSA)			Round-trip (KBAD→LGSA→KBAD)		
	Set-up Time (min)	Run Time (min)	Offload (K lbs.)	Set-up Time (min)	Run Time (min)	Offload (K lbs.)
CMARPS	20	5	0	20	5	250
TAP Tool	< 1	0.05	19.2	< 1	0.06	240

The difference in offload required for the one-way flight is probably due to the climb-fuel required by the B-52. The B-52 can fly 5663 miles on a full tank of fuel. The

total flight distance for this mission is 5428 miles. However, after the B-52 uses 19.2 K lbs. of fuel to climb to altitude, it can only travel 5213 miles with the remaining fuel on board. Depending on the distance the B-52 travels during its climb to altitude, it may have less than 5213 miles remaining in the flight. This is obviously the case in the CMARPS run, since there was no requirement for a refueling point. In our tool, we assume that every aircraft type flies 100 miles during the climb. In this case, the B-52 has 5328 miles remaining when it reaches altitude, which is greater than the number of miles the B-52 can fly with its remaining fuel. The actual number of miles traveled during an aircraft climb to altitude is dependent on the winds and the target altitude.

The second small mission for comparison involves three fighter receiver groups. The groups consist of 6 F-15s, 6 F-16s, and 2 A-10s. Their destination base is Lajes, Greece, while their origin bases are St. Louis, Deluth, and Hurlburt Field respectively. Again, we are interested in the time to run this mission and the total offload required for the receiver groups. Table 15 compares the results of CMARPS with that of the TAP tool.

Table 15: Tool comparison for fighter missions

			Total Offload Required		
	Set-up Time (min)	Run Time (min)	F-16	F-15	A-10
CMARPS	20 - 26	9 - 15	105.1	75.5	28.2
TAP Tool	1	0.05	125.4	80.4	38.5

The total offload required for each aircraft provided by the TAP tool is reasonably close to those provided by CMARPS, according to Maj. Dave Ryer, AMC. The

computation time for the TAP tool is significantly less than that of CMARPS. This computational time benefit is of major interest to AMC.

4.4 Conclusion

4.4.1 Important characteristics demonstrated in thesis

This thesis has demonstrated many new characteristics for a tool designed to assign tankers to receiver groups during a deployment. The previous Quick Look Tool considered tanker availability on a daily basis, and the tankers were not located at specific bases. Our TAP tool considers tanker availability throughout the deployment timeframe. Also, the locations of the refueling points and tankers are used to determine more accurate assignments. The TAP tool takes into consideration the requirement that some receiver groups need escort. Additionally, the previous tool assigned only one tanker to each refueling point. Due to the flexibility in the number of aircraft in each group, our tool allows more than one tanker to satisfy the fuel requirements at a single refueling point.

Our TAP tool uses a form of goal programming to determine the mission evaluation during the search process. We consider the total distance traveled by all tankers in the mission, tanker reuse, and the time at which the receiver groups arrive at their destination. Future research can allow the user to change the weights associated with these goals.

Since almost every computer in the Air Force contains Microsoft Office with Excel, this tool is very portable. Also, most users are familiar with how to enter and

manipulate data within Excel spreadsheets. This increases the usability of the tool and allows new personnel to use the tool with minimal training.

In addition to inputting the receiver group's point of origin and destination, the user may input a waypoint for each receiver group. This waypoint is a location the receiver group travels to before proceeding to the destination base. The waypoint allows pilots to fly alternate routes other than the great circle routes from origin to destination. Many pilots dislike flying over the northern portion of the earth, since emergency landing areas are scarce. The waypoint option allows AMC to generate more realistic missions for deployments.

4.4.2 Problems with the TAP Tool

The TAP tool does not account for the tanker's change in speed during the mission. Normally, tankers decrease their cruise speed during refueling. Tankers then match the speed of fighter aircraft during the escort phase. A calculation would be required to incorporate these changes in speed to determine any modifications to the distance a tanker can travel. This can be accomplished during the preprocessing phase of the program when the decision variable alternatives are determined.

Additionally, every tanker assigned to a refueling point for a fighter receiver group escorts that group to the next critical point. The next critical point may be the next refueling point or the receiver group's destination. This is more restrictive than in an actual deployment. First, fighter groups only need escort over open water, so we need to use a database consisting of land formation locations. Second, although multiple tankers may be needed to satisfy the fighter group's fuel requirements at the refueling point, only

one tanker normally continues to escort the group. To incorporate this in the tool, tankers that return to their base after the refueling should first satisfy a refueling point for a fighter group. Then, an additional decision variable should be created for the tanker escorting the fighter group.

The previous quick look tool uses a third order polynomial equation to determine the fuel burned by an aircraft for a period of flight distance. This method determines fuel burned based on a given flight distance. The TAP tool determines flight distances based on fuel flow. Because of this discrepancy, TAP does not use the *fuelburn* functions in the code. However, the polynomial coefficients for each aircraft type and *fuelburn* functions remain in the code for future implementation.

Although we employ a heuristic to generate a solution, the computation time is polynomial in nature. Each refueling point creates 15 alternatives for each tanker within range. If only one base is within range with 15 tankers stationed there, then this one additional refueling point creates 225 new alternatives. Each alternative, when evaluated during one iteration of TS, makes approximately $100*n$ calculations, where n is the number of refueling points in the mission plan. Therefore, the total number of calculations performed during one iteration is at least $(225*n)(150*n) = (33750*n^2)$. This number doubles in size for each additional tanker base that can service a refueling point. We see that increasing the number of refuelings or tanker bases dramatically increases the calculations being performed.

One way of decreasing this computation time is to decrease the candidate list size. This is the number of neighbors considered at each iteration of the TS. Currently, the user may choose to look at a fraction of the candidate list by inputting a skip number

greater than one; however, there is no logic to choosing which fraction of the candidate list to consider. Future research could develop a means to use the current DV values to determine which part of the candidate list to consider.

Chapter 5.

5.1 Recommendations

The user interface for this tool is very generic. Future development of the tool should include toolbars providing the user with easier methods to input receiver group data, goal programming parameters, and tanker base information. The output data could also be formatted to the user's specifications.

The code for this tool is written in Visual Basic for Applications (VBA) within Excel. In order to run a different deployment, the current set of receiver groups must be replaced with a new set. This includes the aircraft type, number of aircraft, base of origin and destination, RLD and RDD. It would be nice to have the VBA program refer to another workbook containing the list of receiver groups. In this way, the tool would be independent of the input data and would only contain the aircraft performance and tanker location data.

Although most users are comfortable using Excel worksheets to manipulate data, modifying the code to Java would be an improvement. The Java code can be written to import the receiver group data, which the user inputs through an Excel interface. Java is platform independent and is object oriented. This object orientation could increase the manageability of the code and possibly decrease the computation time.

Due to the computation time required for a single run with a small number of iterations, it would be nice to allow the user to stop and restart the tool at any point of the search process. The user would have access to the initial, best, and current mission plans

and would also be able to determine the number of iterations the search process should continue to make once the tool is restarted.

The tool developed in this research uses a constant tabu tenure. Instead, a reactive approach would allow the tenure to change depending on the quality of the missions generated. TS can become trapped in a local optimal section of the solution space because the tenure is not large enough to limit returning to previous similar areas.

The tool currently allows the user to input one waypoint for each receiver group. Future research could add the capability to define this routing further by inputting more waypoints. Furthermore, the additional waypoints could be used along with code development to account for restricted airspace.

Other specific characteristics could further enhance the code. These include adding current characteristics of winds at several locations across the globe, implementing changes in speeds for tankers during aerial-refueling and escort, and adding base resource limitations to the constraints.

Appendix A

```
'Module:      Schedule
'Author:      Lt. Shay R. Capehart, USAF AFIT/ENS
'Last Updated: 24 Feb 00
'Function:    This module contains the primary code for the Quick Look Tanker Deployment
'             Tool. 'Sub Schedule()' is run when the "Schedule" button located on the
'             INPUT sheet is pushed

Dim RPcount As Integer
Dim rp(400, 10) As Variant


Sub Schedule()


' Initialize some variables.  This can be removed if all variables are
' reset or destroyed after this Sub is run.
RPcount = 0
DVcount = 1
numTime = 15    ' Number of increments for take-off tanker times.
getTime = Time

'Define Output Arrays:
'       missplan - mission planning data (Output to MISSION PLAN sheet)
'       recqmts - receiver refueling requirements data
'       sched - current mission schedule
'       initSol - best mission schedule
'       bestSol - initial mission schedule
Dim missplan(200, 15), recrqmts(200, 15), rpvals(200, 40) As Variant
Dim rpdist(200, 15), tankBases(25, 5), DV(50000, 30), RPindex(200) As Variant
Dim sched(200, 15), initSol(200, 15), bestSol(200, 15), tempVar(200, 15) As Variant
Dim CloseLook(60) As Variant


'SECTION I.  INPUT DATA

'i.  Assign user input from INPUT worksheet to data array 'rdata'
'   This assignment allows the user to input as many as 200 mission plans
'   For data processing the number of mission plans entered by the user
'   is counted and assigned to variable 'missions'

rdata = Sheets("INPUT").Range("A7:P207")
missions = Application.Count(Sheets("INPUT").Range("C7:C207"))


'ii.  Assign user input wind data from WINDS worksheet to data array 'winds'
'winds = Sheets("WINDS").Range("A7:B207")

'iii.  Assign ICAO data list in the DISTCALC worksheet to data array 'ICAOlist'  This assignment allows
'    the user to maintain a list of up to 4000 airbases.  The number of bases currently listed is counted
'    and assigned to the variable 'bases'
ICAOlist = Sheets("DISTCALC").Range("A1:F4000")
bases = Application.Count(Sheets("DISTCALC").Range("D2:D4000")) 'Number of bases currently listed on DISTCALC sheet

'iv.  Assign the bases containing tankers to the data array 'tankBases'

For q = 2 To bases + 1
   If ICAOlist(q, 6) > 0 Then
      tankBases(p, 1) = ICAOlist(q, 1)   ' Base ID
      tankBases(p, 2) = ICAOlist(q, 2)   ' Base Name
      tankBases(p, 3) = ICAOlist(q, 4)   ' Base Latitude
      tankBases(p, 4) = ICAOlist(q, 5)   ' Base Longitude
      tankBases(p, 5) = ICAOlist(q, 6)   ' # of tankers at Base
      numTbases = numTbases + 1          ' Increment the number of bases with tankers.
```

```
      p = p + 1
    End If
Next q

'v.  Assign receiver aircraft performance data to data array 'receivers'  This assignment allows the user
'    to maintain data on up to 32 receiver aircraft.  The number of aircraft currently listed is counted and
'    assigned to the variable 'rectype'
receivers = Sheets("AIRCRAFT PERFORMANCE").Range("A5:P37")          'By this definition up to 32 types of receivers may be
listed on
rectype = Application.Count(Sheets("AIRCRAFT PERFORMANCE").Range("B5:B37"))   'Number of receivers currently listed on
AIRCRAFT PERFORMANCE sheet

'vi.  Assign tanker aircraft performance data to data array 'tankers'  This assignment allows the user
'    to maintain data on up to 20 refueling tanker aircraft.  The number of aircraft currently listed is counted and
'    assigned to the variable 'tnkrtype'
tankers = Sheets("AIRCRAFT PERFORMANCE").Range("A41:AC61")
tnkrtype = Application.Count(Sheets("AIRCRAFT PERFORMANCE").Range("B41:B60"))

'vii. Manipulate the input data.

For i = 1 To missions
   ind = Find(rdata(i, 2), Sheets("AIRCRAFT PERFORMANCE").Range("A5:A4000"), rectype)
   n = 1
   While Sheets("INPUT").Cells(i + 6, 3) > receivers(ind, 2)

      Sheets("INPUT").Range(Cells(i + 6, 1), Cells(i + 6, 15)).Copy
      Sheets("INPUT").Paste Destination:=Sheets("INPUT").Range(Cells(missions + 7, 1), Cells(missions + 7, 15))
      Sheets("INPUT").Cells(i + 6, 3) = Sheets("INPUT").Cells(i + 6, 3) - receivers(ind, 2)
      Sheets("INPUT").Cells(missions + 7, 3) = receivers(ind, 2)
      Sheets("INPUT").Cells(missions + 7, 1) = Sheets("INPUT").Cells(missions + 7, 1) & n
      n = n + 1
      missions = missions + 1
   Wend
Next i
rdata = Sheets("INPUT").Range("A7:P207")
missions = Application.Count(Sheets("INPUT").Range("C7:C207"))


'SECTION II.  MISSION PLAN WORKSHEET

icount = 2
RPindex(1) = 1


For i = 1 To missions
'i.  Tansfer the Sortie ID from user input to mission plan data array 'missplan'
   missplan(i, 1) = rdata(i, 1)

'ii. Identify the latitude and longitude of user input origin and destination airbases and assign
'    to mission plan data array 'missplan'
   ind = Find(rdata(i, 4), Sheets("DISTCALC").Range("A1:E4000"), bases)
   If ind = -1 Then
      missplan(i, 2) = "N/A"
      missplan(i, 3) = "N/A"
   Else
      missplan(i, 2) = Sheets("DISTCALC").Cells(ind, 4)
      missplan(i, 3) = Sheets("DISTCALC").Cells(ind, 5)
   End If
   ind = Find(rdata(i, 5), Sheets("DISTCALC").Range("A1:E4000"), bases)
   If ind = -1 Then
      missplan(i, 4) = "N/A"
      missplan(i, 5) = "N/A"
   Else
      missplan(i, 4) = Sheets("DISTCALC").Cells(ind, 4)
      missplan(i, 5) = Sheets("DISTCALC").Cells(ind, 5)
   End If
   If rdata(i, 14) <> 0 Then
      missplan(i, 14) = rdata(i, 14)
```

```
    missplan(i, 15) = rdata(i, 15)
  End If

'iii. Compute the flight window
  missplan(i, 6) = rdata(i, 13) - rdata(i, 12)

'iv. Calculate the flight distance
  If rdata(i, 6) > 0 Then
    missplan(i, 7) = rdata(i, 6)
  ElseIf rdata(i, 14) = 0 Then
    missplan(i, 7) = GreatCircleDistance(missplan(i, 2), missplan(i, 3), missplan(i, 4), missplan(i, 5))
  Else
    missplan(i, 7) = GreatCircleDistance(missplan(i, 2), missplan(i, 3), missplan(i, 14), missplan(i, 15))
    ' Distance to waypoint.
    rpdist(i, 2) = missplan(i, 7)
    ' Add distance from waypoint to destination to get total route distance.
    missplan(i, 7) = missplan(i, 7) + GreatCircleDistance(missplan(i, 14), missplan(i, 15), missplan(i, 4), missplan(i, 5))
  End If

'v.  Referencing the AIRCRAFT PERFORMANCE worksheet, the escort requirement and number of
'    receivers per tanker values are assigned to the missplan data array
  arcft = Find(rdata(i, 2), receivers, rectype)
  missplan(i, 8) = receivers(arcft, 9)
  missplan(i, 9) = receivers(arcft, 2)

'vi. Using the latitudes and longitudes of the origin and destination bases, the mean true
'    course of the receiver is calculated. The mean true course, air speed, wind direction,
'    and wind velocity are then used to calculate the receiver ground speed. Then we set
'    receiver flight duration = great circle distance / ground speed.
  ' TC = TrueCourse(missplan(i, 7), missplan(i, 2), missplan(i, 3), missplan(i, 4), missplan(i, 5))
  ' grdspd = GroundSpeed(rdata(i, 7), TC, winds(i, 1), winds(i, 2))
  missplan(i, 10) = missplan(i, 7) / rdata(i, 7)
  missplan(i, 11) = rdata(i, 7)

'SECTION III. OFFLOAD CALCULATIONS
'The following 'For' loop performs the OFFLOAD CALC Calculations

  recrqmts(i, 1) = rdata(i, 1)   ' Sortie ID
  recrqmts(i, 2) = rdata(i, 11)  ' Total Fuel Capacity
  recrqmts(i, 3) = rdata(i, 7)   ' True air speed
  recrqmts(i, 4) = missplan(i, 11)   ' Ground Speed
  recrqmts(i, 5) = receivers(arcft, 7)  ' Climb Fuel
  recrqmts(i, 6) = receivers(arcft, 4)  ' Fuel Flow
  recrqmts(i, 7) = receivers(arcft, 6)  ' Fuel Reserve
  k1 = receivers(arcft, 12)  ' Four coefficients used to find the fuel burn rate
  k2 = receivers(arcft, 13)
  k3 = receivers(arcft, 14)
  k4 = receivers(arcft, 15)

  temp = rdata(i, 3) * (recflburn(missplan(i, 7), recrqmts(i, 4), recrqmts(i, 2), recrqmts(i, 7), rdata(i, 9), rdata(i, 10), recrqmts(i, 5), k1,
k2, k3, k4) - (recrqmts(i, 2) - recrqmts(i, 7)))
  If temp < 0 Then
    recrqmts(i, 8) = 0  ' Temp Slot
  Else
    recrqmts(i, 8) = temp   ' Temp Slot
  End If

  ' Total fuel required for trip
  recrqmts(i, 9) = recflburn(missplan(i, 7), recrqmts(i, 4), recrqmts(i, 2), recrqmts(i, 7), rdata(i, 9), rdata(i, 10), recrqmts(i, 5), k1, k2,
k3, k4)

  ' Distance between RPs
  recrqmts(i, 10) = (recrqmts(i, 2) - recrqmts(i, 7)) * recrqmts(i, 4) / recrqmts(i, 6)

'SECTION IV.  Calculations for the refueling points.

  rpvals(i, 1) = rdata(i, 1)     ' Sortie ID
  rpvals(i, 2) = missplan(i, 2)  ' Origin Latitude
```

```
rpvals(i, 3) = missplan(i, 3)  ' Origin Longitude
rpvals(i, 4) = missplan(i, 4)  ' Destination Latitude
rpvals(i, 5) = missplan(i, 5)  ' Destination Longitude
rpvals(i, 6) = recrqmts(i, 10)  ' Distance between RPs

' Distance required for entire trip.
fuelReq = recrqmts(i, 5) + (missplan(i, 7) - 100) * recrqmts(i, 6) / recrqmts(i, 4)

' Some aircraft just get refueled right away after take-off.  This will last the rest of the trip.
' Let's assume 100 miles into the flight.

' If distance required is greater than distance between refueling points.
If fuelReq > recrqmts(i, 2) - recrqmts(i, 7) Then
   ' If dist between refueling points is greater than total dist...
   If rpvals(i, 6) > missplan(i, 7) Then
      ' set dist to first refueling equal to 20 miles.
      rpdist(i, 1) = 100
   ' else if aircraft need escort...
   ElseIf missplan(i, 8) = "Y" Then
      '1st ref dist = dist between RPs-(climb fuel*ground speed/fuel flow)
      rpdist(i, 1) = rpvals(i, 6) - (recrqmts(i, 5) * recrqmts(i, 4) / recrqmts(i, 6))
   ' else if input sheet has 1st refueling point overide...
   ElseIf rdata(i, 16) > 0 Then
      ' set 1st refueling point to user input.
      rpdist(i, 1) = rdata(i, 16)
   Else
      rpdist(i, 1) = rpvals(i, 6) - (recrqmts(i, 5) * recrqmts(i, 4) / recrqmts(i, 6))
   End If
End If

' Number of additional legs needed
rpvals(i, 7) = Ceiling((missplan(i, 7) - rpdist(i, 1)) / rpvals(i, 6) - 1)

' Distance to first RP
rpvals(i, 8) = rpdist(i, 1)

' Get information for missions that need refuelings.
If rpvals(i, 8) > 0 Then    ' If the distance to first RP is greater than 0.

   ' If there is no waypoint waypoint...
   If rpdist(i, 2) = 0 Then
      ' Path azymuth from origin to destination.
      rpvals(i, 9) = getAz(rpvals(i, 2), rpvals(i, 3), rpvals(i, 4), rpvals(i, 5))

      ' Latitude for the first refueling point
      rpvals(i, 10) = getLat(rpvals(i, 2), rpdist(i, 1), rpvals(i, 9))

      ' Longitude for the first refueling point
      rpvals(i, 11) = getLong(rpvals(i, 3), rpdist(i, 1), rpvals(i, 9), rpvals(i, 10))

   ' elseif the 1st RP is before the waypoint...
   ElseIf rpvals(i, 8) < rpdist(i, 2) Then
      ' Path azymuth from origin to waypoint.
      rpvals(i, 9) = getAz(rpvals(i, 2), rpvals(i, 3), rdata(i, 14), rdata(i, 15))

      ' Latitude for the first refueling point
      rpvals(i, 10) = getLat(rpvals(i, 2), rpdist(i, 1), rpvals(i, 9))

      ' Longitude for the first refueling point
      rpvals(i, 11) = getLong(rpvals(i, 3), rpdist(i, 1), rpvals(i, 9), rpvals(i, 10))

   ' else the 1st RP is after the waypoint
   Else
      ' Path azymuth from waypoint to destination.
      rpvals(i, 9) = getAz(rdata(i, 14), rdata(i, 15), rpvals(i, 4), rpvals(i, 5))

      dist = rpdist(i, 1) - rpdist(i, 2)
      ' Latitude for the first refueling point
```

```
    rpvals(i, 10) = getLat(rdata(i, 14), dist, rpvals(i, 9))

    ' Longitude for the first refueling point
    rpvals(i, 11) = getLong(rdata(i, 15), dist, rpvals(i, 9), rpvals(i, 10))
End If

' Build the DV alternatives by finding the bases within range.
baseWithinRange = False
RPoffloadSet = False
numTankersAssignedToRP = 1
' Continue this loop which adds tanker to the refueling point, starting
' with one, until the RP is satisfied.
While baseWithinRange = False

    ' Put this information into the RP list.
    Call buildRPs(rdata(i, 1), 1, rpvals(i, 10), rpvals(i, 11), numTankersAssignedToRP)

    For w = 1 To numTankersAssignedToRP
    For q = 1 To numTbases  ' For each base with tankers.
        a = 0
        b = 0
        c = 0
        ' First make temp1 the total distance the tanker can travel.
        temp1 = (tankers(3, 5) - tankers(3, 6)) * tankers(3, 3) / tankers(3, 4)
        ' If RG needs escort, then add the distance between RPs.
        If missplan(i, 8) = "Y" Then
            ' If there are more refueling points for this RG, then the
            ' tanker must escort this group the entire distance between
            ' refuelings "rpvals(i,6)".
            If rpvals(i, 7) > 0 Then
                c = rpvals(i, 6)
                temp1 = temp1 - c
            ' If there are no more refuelings, then the tanker will
            ' escort the fighters to their destination.
            Else
                ' c equals total flight distance minus distance to 1st refueling point.
                c = missplan(i, 7) - rpdist(i, 1)
                temp1 = temp1 - c
            End If
        End If

        ' If there are more refuelings...
        If rpvals(i, 7) > 0 Then
            ' find the coordinates for the next RP
            If rpdist(i, 2) = 0 Then
                azym = getAz(rpvals(i, 10), rpvals(i, 11), rpvals(i, 4), rpvals(i, 5))
                rpvals(i, 10 + 2) = getLat(rpvals(i, 10), rpvals(i, 6), azym)
                rpvals(i, 11 + 2) = getLong(rpvals(i, 11), rpvals(i, 6), azym, rpvals(i, 10 + 2))
            ElseIf rpdist(i, 1) + rpvals(i, 6) < rpdist(i, 2) Then
                azym = getAz(rpvals(i, 10), rpvals(i, 11), rdata(i, 14), rdata(i, 15))
                rpvals(i, 10 + 2) = getLat(rpvals(i, 10), rpvals(i, 6), azym)
                rpvals(i, 11 + 2) = getLong(rpvals(i, 11), rpvals(i, 6), azym, rpvals(i, 10 + 2))
            Else
                azym = getAz(rdata(i, 14), rdata(i, 15), rpvals(i, 4), rpvals(i, 5))
                dist = (rpdist(i, 1) + rpvals(i, 6)) - rpdist(i, 2)
                rpvals(i, 10 + 2) = getLat(rdata(i, 14), dist, azym)
                rpvals(i, 11 + 2) = getLong(rdata(i, 15), dist, azym, rpvals(i, 10 + 2))
            End If
        End If

        ' Subtract from that the distance to this refueling point
        ' and back.
        If missplan(i, 8) = "Y" Then
            If rpvals(i, 7) > 0 Then
                a = GreatCircleDistance(rpvals(i, 10), rpvals(i, 11), tankBases(q, 3), tankBases(q, 4))
                b = GreatCircleDistance(rpvals(i, 10 + 2), rpvals(i, 11 + 2), tankBases(q, 3), tankBases(q, 4))
                temp1 = temp1 - a - b
            Else
```

```
            a = GreatCircleDistance(rpvals(i, 10), rpvals(i, 11), tankBases(q, 3), tankBases(q, 4))
            b = GreatCircleDistance(rpvals(i, 4), rpvals(i, 5), tankBases(q, 3), tankBases(q, 4))
            temp1 = temp1 - a - b
          End If
        Else
          a = GreatCircleDistance(rpvals(i, 10), rpvals(i, 11), tankBases(q, 3), tankBases(q, 4))
          b = a
          temp1 = temp1 - a - b
        End If
        ' Calculate fuel needed at this refueling point: # aircraft*(total dist-dist to first RP)*fuel flow/TAS)
        ' This first temp2 is the amount of fuel needed to finish the total distance.
        ' This is mainly meant for the tankers that get just enough fuel to get to
        ' their destination.
        temp2 = (Application.Min(rdata(i, 3) * (missplan(i, 7) - rpvals(i, 8)) * recrqmts(i, 6) / recrqmts(i, 3), rdata(i, 3) * (recrqmts(i,
5) + (rpvals(i, 8) * recrqmts(i, 6) / recrqmts(i, 3))))) / numTankersAssignedToRP
        ' If the sortie is fighters or there are more refuelings
        ' temp2 represents a full refueling.
        If missplan(i, 8) = "Y" Or rpvals(i, 7) > 0 Then temp2 = (rdata(i, 3) * (recrqmts(i, 2) - recrqmts(i, 7))) /
numTankersAssignedToRP
        ' If the receiver group gets refueled imediately after take-off, then
        ' temp2 is climb fuel.
        If rpvals(i, 8) = 100 Then temp2 = (rdata(i, 3) * recrqmts(i, 5)) / numTankersAssignedToRP

        ' Offload needed at this RP
        If numTankersAssignedToRP = 1 And RPoffloadSet = False Then
          offload = temp2
          RPoffloadSet = True
        End If
        ' How much this amount relates to in terms of distance for the tanker
        temp2 = temp2 * tankers(3, 3) / tankers(3, 4)

        ' Compare values to see if the tanker can arrive at the RP with
        ' enough fuel for the receivers.
        If temp1 - temp2 > 0 Then
          ' Is there a tanker close enough?
          baseWithinRange = True
          ' For each tanker at that base
          For k = 1 To tankBases(q, 5)
            ' For each time increment
            For p = 1 To numTime
              DV(DVcount, 1) = rdata(i, 1)      ' Sortie ID
              DV(DVcount, 2) = 1                ' RP number
              DV(DVcount, 3) = tankBases(q, 1)  ' Base ID
              DV(DVcount, 4) = tankBases(q, 2)  ' Base name
              ' Total distance for tanker trip.
              DV(DVcount, 5) = a + b + c
              ' Fuel available for offload at the RP.
              DV(DVcount, 6) = temp1 * tankers(3, 4) / tankers(3, 3)
              DV(DVcount, 7) = 1 + 5 * (p - 1)  ' Take-off time
              DV(DVcount, 8) = k                ' Tanker number
              DV(DVcount, 12) = a               ' Distance from base to RP
              DVcount = DVcount + 1             ' Increment the total number of decision variables.
            Next p
          Next k
        End If
      Next q

      If baseWithinRange = True Then
        ' Record the number at which the next RP decision variables count.
        RPindex(icount) = DVcount
        icount = icount + 1
      End If

      ' Increment the number of tankers for this RP if there are no tankers capable
      ' of refueling this RP.
      If baseWithinRange = False Then numTankersAssignedToRP = numTankersAssignedToRP + 1

      Next w
```

```
Wend

' Get the remaining refueling points
If rpvals(i, 7) > 0 Then    ' If this sortie has more than 1 refueling point.
  For j = 1 To rpvals(i, 7) ' For each remaining RP.

    baseWithinRange = False
    RPoffloadSet = False
    numTankersAssignedToRP = 1
    While baseWithinRange = False

      ' Add this information to the next spot in the RP index.
      Call buildRPs(rdata(i, 1), j + 1, rpvals(i, 10 + 2 * j), rpvals(i, 11 + 2 * j), numTankersAssignedToRP)

      For w = 1 To numTankersAssignedToRP
      ' Build the decision variables
      For q = 1 To numTbases  ' For each base with tankers.
        a = 0
        b = 0
        c = 0
        temp1 = 0
        ' First make temp1 the total distance the tanker can travel.
        temp1 = (tankers(3, 5) - tankers(3, 6)) * tankers(3, 3) / tankers(3, 4)
        ' If fighters, then subtract the distance between RPs.
        If missplan(i, 8) = "Y" Then
          ' If there are more refueling points for this RG, then the
          ' tanker must escort this group the entire distance between
          ' refuelings "rpvals(i,6)".
          If rpvals(i, 7) > j Then
            c = rpvals(i, 6)
            temp1 = temp1 - c
          ' If there are no more refuelings, then the tanker will
          ' escort the fighters to their destination.
          Else
            c = missplan(i, 7) - rpdist(i, 1) - j * rpvals(i, 6)
            temp1 = temp1 - c
          End If
        End If

        ' If there are more refuelings, find their coordinates
        If rpvals(i, 7) > j Then
          If rpdist(i, 2) = 0 Then
            azym = getAz(rpvals(i, 10 + 2 * j), rpvals(i, 11 + 2 * j), rpvals(i, 4), rpvals(i, 5))
            rpvals(i, 10 + 2 * (j + 1)) = getLat(rpvals(i, 10 + 2 * j), rpvals(i, 6), azym)
            rpvals(i, 11 + 2 * (j + 1)) = getLong(rpvals(i, 11 + 2 * j), rpvals(i, 6), azym, rpvals(i, 10 + 2 * (j + 1)))
          ElseIf rpdist(i, 1) + (j + 1) * rpvals(i, 6) < rpdist(i, 2) Then
            azym = getAz(rpvals(i, 10 + 2 * j), rpvals(i, 11 + 2 * j), rdata(i, 14), rdata(i, 15))
            rpvals(i, 10 + 2 * (j + 1)) = getLat(rpvals(i, 10 + 2 * j), rpvals(i, 6), azym)
            rpvals(i, 11 + 2 * (j + 1)) = getLong(rpvals(i, 11 + 2 * j), rpvals(i, 6), azym, rpvals(i, 10 + 2 * (j + 1)))
          Else
            azym = getAz(rdata(i, 14), rdata(i, 15), rpvals(i, 4), rpvals(i, 5))
            dist = (rpdist(i, 1) + (j + 1) * rpvals(i, 6)) - rpdist(i, 2)
            rpvals(i, 10 + 2 * (j + 1)) = getLat(rdata(i, 14), dist, azym)
            rpvals(i, 11 + 2 * (j + 1)) = getLong(rdata(i, 15), dist, azym, rpvals(i, 10 + 2 * (j + 1)))
          End If
        End If

        ' Subtract the distance to the RP
        If missplan(i, 8) = "Y" Then
          ' If there are more refueling points, then find the
          ' lat and long for the next refueling point.
          If rpvals(i, 7) > j Then
            a = GreatCircleDistance(rpvals(i, 10 + 2 * j), rpvals(i, 11 + 2 * j), tankBases(q, 3), tankBases(q, 4))
            b = GreatCircleDistance(rpvals(i, 10 + 2 * (j + 1)), rpvals(i, 11 + 2 * (j + 1)), tankBases(q, 3), tankBases(q, 4))
            temp1 = temp1 - a - b
          ' Otherwise, subtract the distance to this RP and the distance
          ' to the destination.
```

```
                Else
                  a = GreatCircleDistance(rpvals(i, 10 + 2 * j), rpvals(i, 11 + 2 * j), tankBases(q, 3), tankBases(q, 4))
                  b = GreatCircleDistance(rpvals(i, 4), rpvals(i, 5), tankBases(q, 3), tankBases(q, 4))
                  temp1 = temp1 - a - b
                End If
              Else
                a = GreatCircleDistance(rpvals(i, 10 + 2 * j), rpvals(i, 11 + 2 * j), tankBases(q, 3), tankBases(q, 4))
                b = a
                temp1 = temp1 - a - b
              End If
              ' Fuel needed at this refueling point: # aircraft*(total-reserve fuel)
              ' We are assuming the the only type of receivers that need more than
              ' one refueling are fighters.  Thus, temp2 represents a full refueling.
              If rpvals(i, 7) > j Then
                temp2 = (rdata(i, 3) * (recrqmts(i, 2) - recrqmts(i, 7))) / numTankersAssignedToRP
              Else
                temp2 = (rdata(i, 3) * (missplan(i, 7) - (rpvals(i, 8) + j * rpvals(i, 6))) * recrqmts(i, 6) / recrqmts(i, 3)) /
numTankersAssignedToRP
              End If

              ' Add offload to total RG offload.
              If numTankersAssignedToRP = 1 And RPoffloadSet = False Then
                offload = offload + temp2
                RPoffloadSet = True
              End If
              ' How much this amount relates to in terms of distance for the tanker
              temp2 = temp2 * tankers(3, 3) / tankers(3, 4)
              ' Compare values to see if the tanker can arrive at the RP with
              ' enough fuel for the receivers.
              If temp1 - temp2 > 0 Then
                ' Is there a tanker close enough?
                baseWithinRange = True
                ' For each tanker at that base.
                For k = 1 To tankBases(q, 5)
                  ' For each time increment.
                  For p = 1 To 1
                    DV(DVcount, 1) = rdata(i, 1)      ' Sortie ID
                    DV(DVcount, 2) = j + 1            ' RP number
                    DV(DVcount, 3) = tankBases(q, 1)  ' Base ID
                    DV(DVcount, 4) = tankBases(q, 2)  ' Base name
                    ' Total distance for tanker trip
                    DV(DVcount, 5) = a + b + c
                    ' Fuel available for offload at the RP.
                    DV(DVcount, 6) = temp1 * tankers(3, 4) / tankers(3, 3)
                    DV(DVcount, 7) = p                ' Take-off time
                    DV(DVcount, 8) = k                ' Tanker number
                    DV(DVcount, 12) = a               ' Distance from base to RP
                    DVcount = DVcount + 1             ' Increment the total number of decision variables.
                  Next p
                Next k
              End If
            Next q

            If baseWithinRange = True Then
              ' Record the number at which the next RP decision variables count.
              RPindex(icount) = DVcount
              icount = icount + 1
            End If

            Next w

            ' Increment the number of tankers for this RP if there are no tankers capable
            ' of refueling this RP.
            If baseWithinRange = False Then numTankersAssignedToRP = numTankersAssignedToRP + 1
          Wend
        Next j
      End If
```

```
    ' Display the total offload required for this RG
    Sheets("INPUT").Cells(i + 6, 17) = offload
  End If
Next i

' set the user greedy parameter.
Greedy = Sheets("input").Cells(2, 10)

' If the user has chosen to use the greedy construction heuristic to build the
' initial solution...
If Greedy = "Y" Then
  For i = 1 To RPcount   ' For each RP.
    tempDist = 8000
    tempIndex = 0

    ' Get the index for this mission
    ind = Find(DV(RPindex(i), 1), missplan, missions)

    ' For each DV that can be assigned to this RP.
    For m = RPindex(i) To RPindex(i + 1) - 1
      conflict = False
      ' Choose the base with the smallest round-trip distance for this RP.
      If DV(m, 5) < tempDist Then
        For n = 1 To i  ' For each of the previous RP assignments.
          ' Check to see if the same tanker has already been assigned.
          If DV(m, 3) = sched(n, 3) And DV(m, 8) = sched(n, 6) Then conflict = True
        Next n
        ' If the tanker hasn't already been assigned, record the distance and index for this DV.
        If conflict = False Then

          If DV(m, 2) = 1 Then
            TOA = ((missplan(ind, 7) - rpvals(ind, 8)) / recrqmts(ind, 3)) + DV(m, 7) + DV(m, 12) / tankers(3, 3)
            sched(i, 10) = TOA
            TOD = TOA - (missplan(ind, 7) / recrqmts(ind, 3))
            sched(i, 9) = TOD
            If TOD > 0 Then
              tempDist = DV(m, 5)
              tempIndex = m
            End If
          Else
            tempDist = DV(m, 5)
            tempIndex = m
          End If
        End If
      End If
    Next m
    ' If all tankers capable of servicing this RP have been
    ' assigned at least once.
    If tempIndex = 0 Then
      If DV(m - 1, 2) = 1 Then
        For m = RPindex(i) To RPindex(i + 1) - 1
          conflict = False
          ' Choose the closet base for round-trip distance to this RP.
          If DV(m, 5) < tempDist Then
            For n = 1 To i - 1 ' For each of the previous RP assignments.
              ' Check to see if the same tanker has already been assigned.
              If DV(m, 3) = sched(n, 3) And DV(m, 8) = sched(n, 6) And DV(m, 7) < 10 + sched(n, 5) Then conflict = True
            Next n
            If sched(i - 1, 1) & sched(i - 1, 2) = DV(m, 1) & DV(m, 2) Then
              If sched(i - 1, 5) <> DV(m, 7) Then conflict = True
            End If
            ' If the tanker hasn't already been assigned, record the distance and index for this DV.
            If conflict = False Then

              If DV(m, 2) = 1 Then
                TOA = ((missplan(ind, 7) - rpvals(ind, 8)) / recrqmts(ind, 3)) + DV(m, 7) + DV(m, 12) / tankers(3, 3)
                sched(i, 10) = TOA
                TOD = TOA - (missplan(ind, 7) / recrqmts(ind, 3))
```

```
                    sched(i, 9) = TOD
                    If TOD > 0 Then
                      tempDist = DV(m, 5)
                      tempIndex = m
                    End If
                  Else
                    tempDist = DV(m, 5)
                    tempIndex = m
                  End If
                End If
              End If
            Next m
            ' If there still is no tanker assigned, assign the last
            ' alternative for this DV.
            If tempIndex = 0 Then
              For m = RPindex(i - 1) To RPindex(i) - 1
                If DV(m, 9) = 1 Then
                  tempDist = DV(m, 5)
                  tempIndex = RPindex(i) + (m - RPindex(i - 1))
                End If
              Next m
            End If
          Else
            tempDist = DV(RPindex(i + 1) - DV(m - 1, 2), 5)
            tempIndex = RPindex(i + 1) - DV(m - 1, 2)
          End If
        End If


        DV(tempIndex, 9) = 1
        ' Record this schedule
        sched(i, 1) = DV(tempIndex, 1)
        sched(i, 2) = DV(tempIndex, 2)
        sched(i, 3) = DV(tempIndex, 3)
        sched(i, 4) = DV(tempIndex, 4)
        sched(i, 6) = DV(tempIndex, 8)
        sched(i, 7) = DV(tempIndex, 5)
        sched(i, 11) = DV(tempIndex, 12)

        ' Check to see if this DV is for a RP other than the first one for the receiver group.
        ' If it is, calculate the take-off time.
        If sched(i, 2) <> 1 Then
          a = 0
          temp1 = False
          While temp1 = False
            a = a + 1
            If sched(i - a, 2) < 2 Then temp1 = True
          Wend
          sched(i, 5) = sched(i - a, 8) + (sched(i, 2) - sched(i - a, 2)) * (rpvals(ind, 6) / recrqmts(ind, 3)) - DV(tempIndex, 12) /
tankers(3, 3)
        Else
          sched(i, 5) = DV(tempIndex, 7)
        End If

        ' Determine the time that this refueling will take place
        sched(i, 8) = sched(i, 5) + DV(tempIndex, 12) / tankers(3, 3)

        ' If this is the first refueling point for the receiver group, then
        ' determine the time of arrival (TOA) and departure (TOD) for the
        ' receiver group.
        If sched(i, 2) = 1 Then
          TOA = ((missplan(ind, 7) - rpvals(ind, 8)) / recrqmts(ind, 3)) + sched(i, 8)
          sched(i, 10) = TOA
          TOD = TOA - (missplan(ind, 7) / recrqmts(ind, 3))
          sched(i, 9) = TOD
        End If
      Next i
```

```
' If the greedy construction heuristic is not chosen, then build the initial
' solution by choosing the first available tanker to each RP in the DV list.
Else
   For i = 1 To RPcount
      DV(RPindex(i), 9) = 1
      sched(i, 1) = DV(RPindex(i), 1)
      sched(i, 2) = DV(RPindex(i), 2)
      sched(i, 3) = DV(RPindex(i), 3)
      sched(i, 4) = DV(RPindex(i), 4)
      sched(i, 6) = DV(RPindex(i), 8)
      sched(i, 7) = DV(RPindex(i), 5)
      sched(i, 11) = DV(RPindex(i), 12)

      ' Get the index for this mission
      ind = Find(sched(i, 1), missplan, missions)

      ' Check to see if this DV is for a RP other than the first one for the receiver group.
      ' If it is, calculate the take-off time.
      If sched(i, 2) > 1 Then
         a = 0
         temp1 = False
         While temp1 = False
            a = a + 1
            If sched(i - a, 2) = 1 Then temp1 = True
         Wend
         sched(i, 5) = sched(i - a, 8) + (sched(i, 2) - sched(i - a, 2)) * (rpvals(ind, 6) / recrqmts(ind, 3)) - DV(RPindex(i), 12) / tankers(3,
3)
      Else
         sched(i, 5) = DV(RPindex(i), 7)
      End If

      ' Determine the time that this refueling will take place
      sched(i, 8) = sched(i, 5) + DV(RPindex(i), 12) / tankers(3, 3)

      ' If this is the first refueling point for the receiver group, then
      ' determine the time of arrival (TOA) and departure (TOD) for the
      ' receiver group.
      If sched(i, 2) = 1 Then
         TOA = ((missplan(ind, 7) - rpvals(ind, 8)) / recrqmts(ind, 3)) + sched(i, 8)
         sched(i, 10) = TOA
         TOD = TOA - (missplan(ind, 7) / recrqmts(ind, 3))
         sched(i, 9) = TOD
      End If
   Next i
End If

' Record initial solution to see if it changes
For n = 1 To RPcount
   For i = 1 To 15
      initSol(n, i) = sched(n, i)
   Next i
Next n


numIterations = Sheets("INPUT").Cells(3, 8)
tenure = 1
bestVal = 1000000
' Tabu Search

For n = 1 To RPcount
   CloseLook(n) = True
Next n
infeasible = True

' record the skip number.
modNum = Sheets("INPUT").Cells(4, 10)

For k = 1 To numIterations
```

```
' Display the current iteration.
Sheets("input").Cells(2, 12) = k

Skip = (k Mod modNum) + 1

' Count how many tankers are reused in the current solution.
For q = 1 To RPcount - 1
   p = 1
   goon = True
   While goon = True
      If sched(q, 6) & sched(q, 3) = sched(q + p, 6) & sched(q + p, 3) Then
         If p + q <> i And q <> i Then
            currentReuse = currentReuse + 1
            goon = False
         End If
      End If
      If p = RPcount Then
         goon = False
      Else
         p = p + 1
      End If
   Wend
Next q

i = 1
While i < RPcount + 1

      ' First look at the current solution to count how many conflicts there are.

      conflictNumbers = 0
      currentNegTODs = 0
      If CloseLook(i) = True Then
         If sched(i, 2) = 1 Then

            ' Count how many TODs are less than zero.
            If sched(i, 9) < 0 Then currentNegTODs = currentNegTODs + 1

            h = 0
            While sched(i + h, 1) = sched(i, 1)
               For n = 1 To RPcount
                  If sched(n, 6) & sched(n, 3) = sched(i + h, 6) & sched(i + h, 3) And i + h <> n Then
                     If sched(n, 5) > sched(i + h, 5) Then
                        temp1 = (sched(n, 5) - (3 + sched(i + h, 5) + sched(i + h, 7) / 430))
                     Else
                        temp1 = (sched(i + h, 5) - (3 + sched(n, 5) + sched(n, 7) / 430))
                     End If
                     If temp1 < 0 Then conflictNumbers = conflictNumbers + 1
                  End If
               Next n
               h = h + 1
            Wend
         Else

            For n = 1 To RPcount

               If sched(n, 6) & sched(n, 3) = sched(i, 6) & sched(i, 3) And i <> n Then
                  If sched(n, 5) > sched(i, 5) Then
                     temp1 = (sched(n, 5) - (3 + sched(i, 5) + sched(i, 7) / 430))
                  Else
                     temp1 = (sched(i, 5) - (3 + sched(n, 5) + sched(n, 7) / 430))
                  End If
                  If temp1 < 0 Then conflictNumbers = conflictNumbers + 1
               End If
            Next n
         End If
         ' If there are no problems with the current solution, don't take
         ' a close look at it again.
```

```
            If currentNegTODs = 0 And conflictNumbers = 0 Then CloseLook(i) = False
        End If


    ' Look at each neighbor by changing one RP at a time
    j = RPindex(i) + (modNum - 1)
    While j < RPindex(i + 1)


        ' Save the current solution
        For p = 1 To RPcount
            For n = 1 To 15
                tempVar(p, n) = sched(p, n)
            Next n
        Next p

        ' Change to a new DV for this RP
        sched(i, 1) = DV(j, 1)
        sched(i, 2) = DV(j, 2)
        sched(i, 3) = DV(j, 3)
        sched(i, 4) = DV(j, 4)
        sched(i, 6) = DV(j, 8)
        sched(i, 7) = DV(j, 5)
        sched(i, 11) = DV(j, 12)

        ind = Find(sched(i, 1), missplan, missions)
'XXXXXXXX

        ' If we are changing a RP other than a 1st RP, calculate the
        ' take-off time for this tanker.
        If DV(j, 2) <> 1 Then
            a = 0
            temp1 = False
            While temp1 = False
                a = a + 1
                If sched(i - a, 2) < 2 Then temp1 = True
            Wend
            sched(i, 5) = sched(i - a, 8) + (sched(i, 2) - sched(i - a, 2)) * (rpvals(ind, 6) / recrqmts(ind, 3)) - DV(j, 12) / tankers(3, 3)
            DV(j, 7) = sched(i, 5)
        Else
            sched(i, 5) = DV(j, 7)
        End If

        ' Time of refueling
        sched(i, 8) = sched(i, 5) + sched(i, 11) / tankers(3, 3)


        ' If this is the first refueling point, determine when the receiver
        ' group will arrive at the destination
        If sched(i, 2) = 1 Then

            If sched(i, 1) & sched(i, 2) = sched(i + 1, 1) & sched(i + 1, 2) Then
                sched(i + 1, 8) = sched(i, 8)
            End If

            ' Also calculate the new take-off time for any other tankers
            ' assigned to other refueling points with this same RG.
            n = 1
            While sched(i + n, 1) = sched(i, 1)
                sched(i + n, 5) = sched(i, 8) + (sched(i + n, 2) - sched(i, 2)) * (rpvals(ind, 6) / recrqmts(ind, 3)) - sched(i + n, 11) /
tankers(3, 3)
                n = n + 1
            Wend

            TOA = ((missplan(ind, 7) - rpvals(ind, 8)) / recrqmts(ind, 3)) + sched(i, 8)
            sched(i, 10) = TOA
            TOD = TOA - (missplan(ind, 7) / recrqmts(ind, 3))
            sched(i, 9) = TOD
```

```
End If

' Evaluate this new solution somehow
distPen = 0
temp1 = 0
newConflict = 0
conflictPen = 0
conflictBonus = 0
latePen = 0
earlyBonus = 0
newReuse = 0
reuseBonus = 0
earlyPen = 0
negTODpen = 0
syncPen = 0

' If this is a DV already in the solution, set the penalty to Big-M
' so that it is not chosen for the current move.
If sched(i, 3) & sched(i, 5) & sched(i, 6) = tempVar(i, 3) & tempVar(i, 5) & tempVar(i, 6) Then
   DV(j, 10) = 1000000

Else

   ' penalize for making the receivers take off too early.
   If TOD < 0 Then earlyPen = 500000

   If earlyPen <> 500000 Then

      ' penalize for late receivers
      If sched(i, 2) = 1 Then
         If Max(0, ((TOA) - 24 * missplan(ind, 6))) > 0 Then latePen = 600000
         earlyBonus = 10 * Max(0, ((24 * missplan(ind, 6)) - (TOA)))
      End If

      If latePen <> 600000 Then
         For n = 1 To RPcount
            distPen = distPen + sched(n, 7)
            ' penalize for having the same aircraft take off sooner than 10 hours apart
            If sched(n, 6) & sched(n, 3) = sched(i, 6) & sched(i, 3) And i <> n Then
               If sched(n, 5) > sched(i, 5) Then
                  temp1 = (sched(n, 5) - (3 + sched(i, 5) + sched(i, 7) / 430))
               Else
                  temp1 = (sched(i, 5) - (3 + sched(n, 5) + sched(n, 7) / 430))
               End If
               If temp1 < 0 Then newConflict = newConflict + 1
               If sched(i, 6) & sched(i, 3) <> tempVar(i, 6) & tempVar(i, 3) Then newReuse = newReuse + 1
            End If

            ' penalize for not having multiple first refueling points at the same time
            If sched(n, 2) = 1 And sched(i, 2) = 1 And sched(n, 1) = sched(i, 1) Then
               If sched(n, 8) <> sched(i, 8) Then syncPen = 40000
            End If
         Next n

         ' If this is the first refueling point for a receiver group,
         ' check the other refueling points to make sure you haven't
         ' created a conflict.
         If sched(i, 2) = 1 Then
            h = 0
            While sched(i + h, 1) = sched(i, 1)
               If h > 0 Then
                  ' Take-off Time
                  sched(i + h, 5) = sched(i, 8) + (sched(i + h, 2) - sched(i, 2)) * (rpvals(ind, 6) / recrqmts(ind, 3)) - sched(i + h,
11) / tankers(3, 3)
                  ' Time of refueling
                  sched(i + h, 8) = sched(i + h, 5) + sched(i + h, 11) / tankers(3, 3)
               End If
               For n = 1 To RPcount
```

```
                    If sched(n, 6) & sched(n, 3) = sched(i + h, 6) & sched(i + h, 3) And i + h <> n Then
                      If sched(n, 5) > sched(i + h, 5) Then
                        temp1 = (sched(n, 5) - (3 + sched(i + h, 5) + sched(i + h, 7) / 430))
                      Else
                        temp1 = (sched(i + h, 5) - (3 + sched(n, 5) + sched(n, 7) / 430))
                      End If
                      If temp1 < 0 Then newConflict = newConflict + 1
                    End If
                  Next n
                  h = h + 1
                Wend
              End If

              reuseBonus = 500 * newReuse + 2000 * currentReuse

              ' Reward if the number of conflicts has decreased.
              ' Penalize if there are more conflicts.
              If newConflict < conflictNumbers Then
                conflictBonus = -100000
              ElseIf newConflict > 0 Or conflictNumbers > 0 Then
                conflictPen = 90000
              End If
              If currentNegTODs > 0 Then
                negTODpen = -100000 * currentNegTODs
              End If
            End If
          End If

'         DV(j, 18) = conflictPen
'         DV(j, 19) = conflictBonus
'         DV(j, 20) = distPen
'         DV(j, 21) = reuseBonus
'         DV(j, 22) = earlyPen
'         DV(j, 23) = negTODpen
'         DV(j, 24) = earlyBonus
'         DV(j, 25) = latePen
'         DV(j, 26) = newConflict
'         DV(j, 27) = conflictNumbers
'         DV(j, 28) = CloseLook(i)
'         DV(j, 29) = infeasible

          DV(j, 10) = conflictPen + conflictBonus + distPen - reuseBonus + syncPen + earlyPen + negTODpen + sched(i, 6) -
earlyBonus + latPen

        End If



        ' Return the current solution
        For p = 1 To RPcount
          For n = 1 To 15
            sched(p, n) = tempVar(p, n)
          Next n
        Next p
      j = j + modNum
      Wend

  i = i + 1
  Wend

  ' Check to see if the current solution is now feasible.
  If infeasible = True Then
    infeasible = False
    For i = 1 To RPcount
      If CloseLook(i) = True Then
        infeasible = True
      End If
    Next i
```

```
    If infeasible = False Then tenure = Sheets("INPUT").Cells(2, 8)
  End If

  ' Look at all the neighbors and choose the smallest that isn't tabu
  objectiveVal = 1000000
  For j = 1 To RPcount
    i = RPindex(j) + (modNum - 1)
    While i < RPindex(j + 1)
      'Sheets("input").Cells(4, 12) = i
      If DV(i, 10) <= objectiveVal Then
        If DV(i, 11) < 1 Then
          objectiveVal = DV(i, 10)
          DVmove = i
          moveIndex = j
        End If
        If DV(i, 10) < bestVal And infeasible = False And DV(i, 10) > -50000 Then
          bestVal = DV(i, 10)
          objectiveVal = DV(i, 10)
          DVmove = i
          moveIndex = j
        End If
      End If
    i = i + modNum
    Wend
  Next j



  ' Make the move
  currentObjVal = DV(DVmove, 10)
      For n = RPindex(moveIndex) To RPindex(moveIndex + 1) - 1
        If DV(n, 9) = 1 Then DV(n, 9) = 0
        If DV(n, 3) = DV(DVmove, 3) Then DV(n, 11) = tenure + 1
      Next n
      i = moveIndex
      sched(i, 1) = DV(DVmove, 1)
      sched(i, 2) = DV(DVmove, 2)
      sched(i, 3) = DV(DVmove, 3)
      sched(i, 4) = DV(DVmove, 4)
      sched(i, 6) = DV(DVmove, 8)
      sched(i, 7) = DV(DVmove, 5)
      sched(i, 11) = DV(DVmove, 12)

      ind = Find(sched(i, 1), missplan, missions)

      If sched(i, 2) = 1 Then
        ' Take-off Time
        sched(i, 5) = DV(DVmove, 7)
        ' Time of refueling
        sched(i, 8) = sched(i, 5) + sched(i, 11) / tankers(3, 3)
        h = 1
        While sched(i + h, 1) = sched(i, 1)
          ' Take-off time
          sched(i + h, 5) = sched(i, 8) + (sched(i + h, 2) - sched(i, 2)) * (rpvals(ind, 6) / recrqmts(ind, 3)) - sched(i + h, 11) /
tankers(3, 3)
          ' Time of refueling
          sched(i + h, 8) = sched(i + h, 5) + sched(i + h, 11) / tankers(3, 3)
          h = h + 1
        Wend
        TOA = ((missplan(ind, 7) - rpvals(ind, 8)) / recrqmts(ind, 3)) + sched(i, 8)
        sched(i, 10) = TOA
        TOD = TOA - (missplan(ind, 7) / recrqmts(ind, 3))
        sched(i, 9) = TOD

        If sched(i, 1) & sched(i, 2) = sched(i + 1, 1) & sched(i + 1, 2) Then
          sched(i + 1, 5) = sched(i, 5)
          sched(i + 1, 10) = sched(i, 10)
          sched(i + 1, 9) = sched(i, 9)
```

```
            End If
        End If
    DV(DVmove, 9) = 1

    ' Display the current variable being removed.
    If Sheets("INPUT").Cells(4, 8) = "S" Then
      DV(k, 13) = DV(DVmove, 1) & DV(DVmove, 2) & DV(DVmove, 3)
    Else
      DV(k, 13) = DV(DVmove, 1) & DV(DVmove, 2)
    End If
    ' Display the mission evaulation value for the current iteration.
    DV(k, 14) = DV(DVmove, 10)
    For i = 1 To DVcount - 1
      If DV(i, 11) >= 1 Then DV(i, 11) = DV(i, 11) - 1
    Next i


    ' If this move produces a new best solution, change the bestVal and
    ' record this new solution in the bestSol array.
    If DV(DVmove, 10) = bestVal Then
      For i = 1 To RPcount
        For n = 1 To 15
          bestSol(i, n) = sched(i, n)
        Next n
      Next i
    End If


'Print the; current; solution
'   Sheets("OTHER DATA").Select
'   For p = 1 To RPcount
'     For q = 1 To 15
'       Cells(RPcount * (k - 1) + (p + 1), q) = sched(p, q)
'     Next q
'   Next p

Next k

'Output data to worksheets
Sheets("MISSION PLAN").Select
Range("A2:K207").Clear
For p = 1 To missions
  For q = 1 To 11
    Cells(p + 1, q) = missplan(p, q)
  Next q
Next p
'Range("B2:B500").Select
'Selection.NumberFormat = _
'   "000 ""deg"" 00.0 ""min N"";000 ""deg"" 00.0 ""min S"""
'Range("C2:C500").Select
'Selection.NumberFormat = _
'   "000 ""deg"" 00.0 ""min W"";000 ""deg"" 00.0 ""min E"""
'Range("D2:D500").Select
'Selection.NumberFormat = _
'   "000 ""deg"" 00.0 ""min N"";000 ""deg"" 00.0 ""min S"""
'Range("E2:E500").Select
'Selection.NumberFormat = _
'   "000 ""deg"" 00.0 ""min W"";000 ""deg"" 00.0 ""min E"""
'
Sheets("REFUELING POINTS").Select
Range("A2:V207").Clear
For p = 1 To missions
  For q = 1 To 20
    Cells(p + 1, q) = rpvals(p, q)
  Next q
Next p
Range("B2:B500").Select
```

```
Selection.NumberFormat = _
    "000 ""deg"" 00.0 ""min N"";000 ""deg"" 00.0 ""min S"""
Range("C2:C500").Select
Selection.NumberFormat = _
    "000 ""deg"" 00.0 ""min W"";000 ""deg"" 00.0 ""min E"""
Range("D2:D500").Select
Selection.NumberFormat = _
    "000 ""deg"" 00.0 ""min N"";000 ""deg"" 00.0 ""min S"""
Range("E2:E500").Select
Selection.NumberFormat = _
    "000 ""deg"" 00.0 ""min W"";000 ""deg"" 00.0 ""min E"""
Range("J2:J500").Select
Selection.NumberFormat = _
    "000 ""deg"" 00.0 ""min N"";000 ""deg"" 00.0 ""min S"""
Range("K2:K500").Select
Selection.NumberFormat = _
    "000 ""deg"" 00.0 ""min W"";000 ""deg"" 00.0 ""min E"""
Range("L2:L500").Select
Selection.NumberFormat = _
    "000 ""deg"" 00.0 ""min N"";000 ""deg"" 00.0 ""min S"""
Range("M2:M500").Select
Selection.NumberFormat = _
    "000 ""deg"" 00.0 ""min W"";000 ""deg"" 00.0 ""min E"""
Range("N2:N500").Select
Selection.NumberFormat = _
    "000 ""deg"" 00.0 ""min N"";000 ""deg"" 00.0 ""min S"""
Range("O2:O500").Select
Selection.NumberFormat = _
    "000 ""deg"" 00.0 ""min W"";000 ""deg"" 00.0 ""min E"""
Range("P2:P500").Select
Selection.NumberFormat = _
    "000 ""deg"" 00.0 ""min N"";000 ""deg"" 00.0 ""min S"""
Range("Q2:Q500").Select
Selection.NumberFormat = _
    "000 ""deg"" 00.0 ""min W"";000 ""deg"" 00.0 ""min E"""
Range("R2:R500").Select
Selection.NumberFormat = _
    "000 ""deg"" 00.0 ""min N"";000 ""deg"" 00.0 ""min S"""
Range("S2:S500").Select
Selection.NumberFormat = _
    "000 ""deg"" 00.0 ""min W"";000 ""deg"" 00.0 ""min E"""
'For w = 1 To Max([G2:G200])
'   Range(Cells(2, 8 + 2 * w), Cells(200, 8 + 2 * w)).Select
'   Selection.NumberFormat = _
'       "000 ""deg"" 00.0 ""min N"";000 ""deg"" 00.0 ""min S"""
'   Range(Cells(2, 9 + 2 * w), Cells(200, 9 + 2 * w)).Select
'   Selection.NumberFormat = _
'       "000 ""deg"" 00.0 ""min W"";000 ""deg"" 00.0 ""min E"""
'Next w

'Sheets("OFFLOAD CALC").Select
'Range("A2:J207").Clear
'For p = 1 To missions
'   For q = 1 To 10
'       Cells(p + 1, q) = recrqmts(p, q)
'   Next q
'Next p

Sheets("RP LOCATIONS").Select
Range("A2:J207").Clear
For p = 1 To RPcount
   For q = 1 To 10
      Cells(p + 1, q) = rp(p, q)
   Next q
Next p

Sheets("DECISION VARIABLES").Select
```

```
If Sheets("input").Cells(3, 10) = "Y" Then
   'Range("A2:M40000").Clear
   For p = 1 To DVcount
      For q = 1 To 11
         Cells(p + 1, q) = DV(p, q)
      Next q
      'For q = 18 To 30
      '   Cells(p + 1, q) = DV(p, q)
      'Next q
   Next p
End If


For p = 1 To numIterations
   For q = 13 To 14
      Cells(p + 1, q) = DV(p, q)
   Next q
Next p

Sheets("BEST MISSION PLAN").Select
Call displayResults(bestSol, RPcount)
Sheets("FIRST MISSION PLAN").Select
Call displayResults(initSol, RPcount)
Sheets("LAST MISSION PLAN").Select
Call displayResults(sched, RPcount)
Sheets("BEST MISSION PLAN").Select

getTime = (Time - getTime)
Sheets("Input").Cells(3, 12) = getTime
End Sub

Sub buildRPs(index, num, lat, lng, part)
   RPcount = RPcount + 1
   rp(RPcount, 1) = index
   rp(RPcount, 2) = num
   rp(RPcount, 3) = lat
   rp(RPcount, 4) = lng
   rp(RPcount, 5) = part
End Sub

Sub displayResults(array1 As Variant, n As Integer)
   Range("A2:O400").Clear
   For p = 1 To n
      For q = 1 To 14
         Cells(p + 1, q) = array1(p, q)
      Next q
   Next p
   tankerReuse = 0
   For q = 1 To n - 1
      p = 1
      goon = True
      While goon = True
         If array1(q, 6) & array1(q, 3) = array1(q + p, 6) & array1(q + p, 3) Then
            If p + q <> i And q <> i Then
               tankerReuse = tankerReuse + 1
               goon = False
            End If
         End If
         If p = n Then
            goon = False
         Else
            p = p + 1
         End If
      Wend
   Next q
   Cells(n + 3, 5) = n - tankerReuse
   Cells(n + 3, 4) = "Tankers Used:"
   Cells(n + 4, 5) = Application.Sum(Range(Cells(2, 7), Cells(n + 1, 7)))
```

```
    Cells(n + 4, 4) = "Total Tanker Distance:"
    Cells(n + 5, 5) = Application.Max(Range(Cells(2, 10), Cells(n + 1, 10)))
    Cells(n + 5, 4) = "Latest Receiver Group TOA:"

End Sub
```

```
'Subroutine Name:  setGWdefaults()
'Functionality:    This function sets the default values for the weight
'                  characteristics of receiver aircraft. Including:
'                    1) Minimum Weight Empty
'                    2) Payload Weight
'                    3) Fuel Weight
'                  The default values are set in the AIRCRAFT PERFORMANCE sheet
'Arguments:        None

Sub setdefaults()

missions = Application.Count(Sheets("INPUT").Range("C7:C207"))
rdata = Sheets("INPUT").Range("A7:P207")

rec = Application.Count(Sheets("AIRCRAFT PERFORMANCE").Range("B5:B37"))
receivers = Sheets("AIRCRAFT PERFORMANCE").Range("A5:P37")

For i = 1 To missions
   ind = Find(rdata(i, 2), receivers, rec)
   Sheets("INPUT").Cells(i + 6, 9) = receivers(ind, 10)
   Sheets("INPUT").Cells(i + 6, 10) = receivers(ind, 11)
   Sheets("INPUT").Cells(i + 6, 11) = receivers(ind, 5)
   Sheets("INPUT").Cells(i + 6, 8) = receivers(ind, 8)
   Sheets("INPUT").Cells(i + 6, 7) = receivers(ind, 3)
Next i
End Sub

'Subroutine Name:  setWddefaults
'Functionality:    This function sets the winds to zero
'Arguments:        None

Sub setWddefaults()

missions = Application.Count(Sheets("INPUT").Range("C7:C200"))

For i = 1 To missions
   Sheets("WINDS").Cells(i + 6, 1) = 0
   Sheets("WINDS").Cells(i + 6, 2) = 0
Next i

End Sub
```

```
'Function Name: Ceiling
'Functionality: This function rounds fractional numbers
'              to the next highest integer
'Arguments:     x - The function finds the ceiling of x
'Return Value:  Ceiling - the ceiling of x

Function Ceiling(x)
   temp = CInt(x)
   temp1 = temp - x
   If temp1 > 0 Then
      Ceiling = temp
   Else
      Ceiling = temp + 1
   End If
End Function
Function Max(a, b)
   If a > b Then
      Max = a
   Else
      Max = b
   End If
End Function


Function Floor(x)
   temp = Abs(x)
   temp = Ceiling(temp)
   If x > 0 Then
      Floor = temp - 1
   Else
      Floor = -temp
   End If
End Function


'Function Name: Find
'Functionality: This function determines the row position
'              of a given aircraft in the distcalc or
'              aircraft performance matrices
'Arguments:     item - the name of the aircraft to find
'              list - the matrix to search
'              total - the number of rows in the search matrix
'Return Value:  pos - the row position of the aircraft

Function Find(item, list, total)
   pos = 1
   found = False
      While Not found
         If StrComp(item, list(pos, 1)) = 0 Then
            Find = pos
            found = True
         Else
            pos = pos + 1
         End If
         If pos > total Then
            Find = -1
            found = True
         End If
      Wend
End Function
```

```
'Function Name: fuelburn
'Functionality: This function is used to determine the fuel
'          burned by a given fighter for a period of
'          flight given by: Flight time = Distance/True Air Speed
'          The algorithm assumes a nominal flight altitude and
'          true air speed. The fuel flow is calculated with a third
'          order polynomial model of the fuel flow depending on gross weight.
'          It is assumed that the fighter's fuel is burned down to
'          the fuel reserve level and then completely refueled.
'Arguments:  dist - the distance the fighter will travel
'          tas - the true airspeed the fighter will travel at
'          r - the fighter performance matrix
'          j - the position of the desired fighter in the performance matrix
'Return Value:  totfb - total fuel burned over the flight

Function recflburn(dist, rate, fuelcap, reserve, minwt, cargo, climbf, c1, c2, c3, c4)

  fb = 0
  totfb = climbf
  ff = 0
  mult = 0

  gw = 0
  gwi = fuelcap + minwt + cargo
  maxburn = fuelcap - reserve

  Flighttime = dist / rate
  dt = 0.01

  For t = 1 To Flighttime * 100
    Nar = Ceiling(totfb / maxburn) - 1
    gw = gwi - totfb + Nar * maxburn
    ff = c1 + c2 * gw + c3 * gw * gw + c4 * gw * gw * gw
    fb = ff * dt
    totfb = totfb + fb
  Next t

  recflburn = totfb

End Function

Function tnkrflburn(dist, rrate, trate, rgs, tgs, fuelcap, reserve, minwt, cargo, climbf, ralt, talt, c1, c2, c3, c4, c5, c6, c7, test)

  t = 0
  dt = 0.01
  maxburn = fuelcap - reserve
  fb = 0
  ff = 0
  gw = 0
  gwi = fuelcap + minwt + cargo
  totfb = climbf

  If StrComp(test, "F") = 0 Then
    fltm = 0.5 * dist / rgs + 0.5 * dist / tgs
  ElseIf StrComp(test, "R") = 0 Then
    fltm = dist / rgs
  Else
    fltm = dist / tgs
  End If

  While t < fltm * 100
    If StrComp(test, "F") = 0 Then
      If t * dt < 0.5 * fltm Then
```

```
            alt = ralt
            TAS = rrate
          Else
            alt = talt
            TAS = trate
          End If
        Else
          alt = talt
          TAS = trate
        End If

          gw = gwi - totfb
          fflow = c1 + c2 * alt + c3 * alt * alt + c4 * TAS + c5 * TAS * TAS + c6 * gw + c7 * gw * gw
          fb = fflow * dt
          totfb = totfb + fb

          t = t + 1

    Wend
    tnkrflburn = totfb

End Function
```

```
'Module:    DistCalc
'Function:  This module contains the functions for calculating the great circle
'          distance from the origin to the destination bases


'Function Name: DecDeg
'Functionality: To decode the DDDMM.M format (where D=degrees, M=Minutes) for Latitude
'              and Longitude to degrees.
'Arguments:     Number - Value passed to function in DDDMM.M format
'Return Value:  Temp - the Latitude or Longitude in degrees

Private Function DecDeg(Number)

   num = Abs(Number)   ' Get absolute value of Number to use in Int()

   temp = Int(num / 100) + (num / 100 - Int(num / 100)) / 0.6
                       ' Convert by separating integer degrees from
                       ' minutes portion.  Then divide minutes by 60
                       ' to get fractional degrees and add to integer
                       ' degrees.

   If num > Number Then          ' Check that Temp has same sign (+/-) as Number
      temp = -temp               ' before assigning to return value
   End If
   DecDeg = temp                 ' Assign Temp to function's return value
End Function
Function DegDec(Number)

   num = Abs(Number)

   temp = (Floor(num) * 100) + (60 * (num - Floor(num)))

   If Number < 0 Then temp = -temp

   DegDec = temp
End Function

'Function Name: GreatCircleDistance
'Functionality: To compute great circle distance between two points on Earth.  Points
'              are (Latitude1, Longitude1) and (Latitude2, Longitude2).  This function
'              accepts latitude and longitude in real degrees or in DDDMM.M format.
'Arguments:     latitude1 - origin latitude
'              longitude1 - orgin longitude
'              latitude2 - destination latitude
'              longitude2 - destination longitude
'Return Value:  GreatCircleDistance - the great circle distance

Function GreatCircleDistance(latitude1, longitude1, latitude2, longitude2)

   Deg2Rad = 3.14159265358979 / 180   'Define constants
   Rad2Deg = 180 / 3.14159265358979   'for angle conversions
   NMperDeg = 60

   lat1 = latitude1
   lat2 = latitude2
   long1 = longitude1
   long2 = longitude2

   If (Abs(lat1) > 90) Or (Abs(lat2) > 90) Or (Abs(long1) > 180) Or (Abs(long2) > 180) Then
      lat1 = DecDeg(lat1)   ' Assumes all coordinates are in same
      lat2 = DecDeg(lat2)   ' format. If any are found in DDDMM.M
      long1 = DecDeg(long1) ' format then convert all to degrees.
```

```
    long2 = DecDeg(long2)
  End If

  lat1 = lat1 * Deg2Rad    ' Convert all degrees to radians
  lat2 = lat2 * Deg2Rad
  long1 = long1 * Deg2Rad
  long2 = long2 * Deg2Rad

  temp = Cos(lat1) * Cos(lat2) * Cos(long2 - long1)

  temp = Application.Acos(temp + Sin(lat1) * Sin(lat2)) * Rad2Deg
                          ' Calculated the angle of the great circle
                          ' arc between the two points. Formula
                          ' came from original AMCSAF Distcalc
                          ' spreadsheet. Uses Excel's ACOS().

 GreatCircleDistance = NMperDeg * temp  ' Convert arc degrees to NM and return
End Function
Function getAz(latitude1, longitude1, latitude2, longitude2)

  Deg2Rad = 3.14159265358979 / 180   'Define constants
  Rad2Deg = 180 / 3.14159265358979   'for angle conversions
  NMperDeg = 60

  lat1 = latitude1
  long1 = longitude1
  lat2 = latitude2
  long2 = longitude2
  dist = GreatCircleDistance(lat1, long1, lat2, long2)

  If (Abs(lat2) > 90) Or (Abs(long1) > 180) Or (Abs(long2) > 180) Then
    lat1 = DecDeg(lat1)  ' Assumes all coordinates are in same
    lat2 = DecDeg(lat2)  ' format. If any are found in DDDMM.M
    long1 = DecDeg(long1) ' format then convert all to degrees.
    long2 = DecDeg(long2)
  End If

  lat1 = lat1 * Deg2Rad
  lat2 = lat2 * Deg2Rad
  long1 = long1 * Deg2Rad
  long2 = long2 * Deg2Rad
  dist = dist / NMperDeg
  dist = dist * Deg2Rad

  sinAz = (Cos(lat2) * Sin(long2 - long1) / Sin(dist))
  cosAz = ((Sin(lat2) - (Cos(dist) * Sin(lat1))) / (Sin(dist) * Cos(lat1)))
  If sinAz >= 0 And cosAz >= 0 Then
    temp = Application.Asin(sinAz)
  ElseIf sinAz >= 0 And cosAz < 0 Then
    temp = 3.14159265358979 - Application.Asin(sinAz)
  ElseIf cosAz >= 0 Then
    temp = -Application.Acos(cosAz)
  Else
    temp = -(3.14159265358979 + Application.Asin(sinAz))
  End If
  temp = temp * Rad2Deg
  getAz = temp
End Function
Function getLat(latitude1, distance, azymuth)

  Deg2Rad = 3.14159265358979 / 180   'Define constants
  Rad2Deg = 180 / 3.14159265358979   'for angle conversions
  NMperDeg = 60

  lat1 = latitude1
  dist = distance
  Az = azymuth
```

```
  If (Abs(lat1) > 90) Then lat1 = DecDeg(lat1)

  lat1 = lat1 * Deg2Rad
  dist = dist / NMperDeg
  dist = dist * Deg2Rad
  Az = Az * Deg2Rad

  temp = Application.Acos(Sin(lat1) * Cos(dist) + Cos(lat1) * Sin(dist) * Cos(Az))
  temp = temp * Rad2Deg
  temp = 90 - temp
  getLat = DegDec(temp)
End Function
Function getLong(longitude1, distance, azymuth, latitudeRP)

  Deg2Rad = 3.14159265358979 / 180   'Define constants
  Rad2Deg = 180 / 3.14159265358979   'for angle conversions
  NMperDeg = 60

  long1 = longitude1
  dist = distance
  Az = azymuth
  latRP = latitudeRP

  If (Abs(long1) > 90) Or (Abs(latRP) > 90) Then
    long1 = DecDeg(long1)
    latRP = DecDeg(latRP)
  End If

  dist = dist / NMperDeg
  dist = dist * Deg2Rad
  Az = Az * Deg2Rad
  latRP = latRP * Deg2Rad
  long1 = long1 * Deg2Rad

  temp = Application.Asin(Sin(dist) * Sin(Az) / Cos(latRP))
  temp = temp + long1
  temp = temp * Rad2Deg
  If temp > 180 Then temp = temp - 360
  getLong = DegDec(temp)
End Function

Function TrueCourse(dist, latitude1, longitude1, latitude2, longitude2)

  Deg2Rad = 3.14159265358979 / 180   'Define constants
  Rad2Deg = 180 / 3.14159265358979   'for angle conversions
  p = 3.1415926535897

  la1 = latitude1
  lg1 = longitude1
  la2 = latitude2
  lg2 = longitude2

  If (Abs(la1) > 90) Or (Abs(la2) > 90) Or (Abs(lg1) > 180) Or (Abs(lg2) > 180) Then
    la1 = DecDeg(la1)  ' Assumes all coordinates are in same
    la2 = DecDeg(la2)  ' format. If any are found in DDDMM.M
    lg1 = DecDeg(lg1)  ' format then convert all to degrees.
    lg2 = DecDeg(lg2)
  End If

  la1 = la1 * Deg2Rad    ' Convert all degrees to radians
  la2 = la2 * Deg2Rad
  lg1 = lg1 * Deg2Rad
  lg2 = lg2 * Deg2Rad
  D = (dist / 60) * Deg2Rad

  H1 = Application.Acos((Sin(la2) - Sin(la1) * Cos(D)) / (Sin(D) * Cos(la1)))
  H2 = Application.Acos((Sin(la1) - Sin(la2) * Cos(D)) / (Sin(D) * Cos(la2)))
```

```
  If Sin(lg2 - lg1) < 0 Then
    Hi1 = H1
  Else
    Hi1 = 2 * p - H1
  End If

  If Sin(lg1 - lg2) < 0 Then
    Hi2 = H2
  Else
    Hi2 = 2 * p - H2
  End If

  If Hi2 >= p Then
    Hi2 = Hi2 - p
  Else
    Hi2 = Hi2 + p
  End If
  TrueCourse = (Hi1 + Hi2) / 2 * Rad2Deg

End Function

Function GroundSpeed(TAS, TC, Wd, Wv)

  Deg2Rad = 3.14159265358979 / 180   'Define constants
  Rad2Deg = 180 / 3.14159265358979   'for angle conversions

  TCr = TC * Deg2Rad

  Wdr = Wd * Deg2Rad
  DCA = Application.Asin((Wv / TAS) * Sin(Wdr - TCr))
  GroundSpeed = TAS * Cos(DCA) - Wv * Cos(Wdr - TCr)

End Function
```

Bibliography

Battiti, R. "Reactive search: Toward self-tuning heuristics," Modern Heuristic Search Methods, Rayward-Smith (ed.), John Wiley and Sons Ltd: 61-83, 1996.

Battiti, R. and G. Tecchiolli. "The reactive tabu search," ORSA Journal on Computing, 6(2): 126-140, 1994.

Ben-Daya, M. and M. Al-Fawzan. "A tabu search approach for the flow shop scheduling problem," European Journal of Operational Research, v109, 88-95, 1998.

Committee on the Next Decade of Operations Research (CONDOR) "Operations Research: The Next Decade," Operations Research, Vol. 36: 619-637, 1988.

Congress of the United States Congressional Budget Office. Modernizing the Aerial Tanker Fleet: Prospects for Capacity, Timing, Cost. Washington: Congressional Budget Office, 1985.

Glover, F. "Future Paths for Integer Programming and Links to Artificial Intelligence," Computers and Operations Research, Vol. 13: 533-539, 1986.

Glover, F. "Tabu Search: A Tutorial," Interfaces, Vol. 20: 74-94, 1990.

Glover, F. and M. Laguna. Tabu Search. Boston: Kluwer Academic Publishers, 1997.

Hostler, H. Air Refueling Tanker Scheduling. Air Force Institute of Technology, Wright-Patterson AFB, OH, 1987.

Logicon. "Combined Mating And Ranging Planning System Overview." Slide presentation, Information Technology Group, Logicon Inc, 1996.

Russina, B., Ruthsatz, B., and Russ. "The Quick Look Tool for Tanker Deployment." Center for Optimization and Semantic Control. Washington University, St. Louis, MO, 1999.

Silver, E., Vidal R., and D. Werra. "A tutorial on heuristic methods," Europen Journal of Operational Research, Vol. 5, 153-162, 1980.

Woodruff, D.L., and Zemel, E., "Hashing vectors for tabu search," Annals of Operations Research, Vol. 41: 123-137, 1993.

Zanakis, S.H. "Heuristic 'Optimization': Why, When, And How To Use It," Interfaces, Vol. 11, 1981.